VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

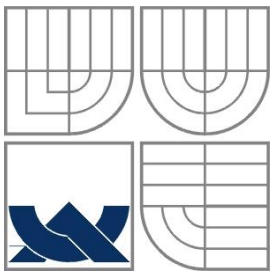
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SOFTWARE PRO PODPORU ONLINE HRY ENTROPIA UNIVERSE

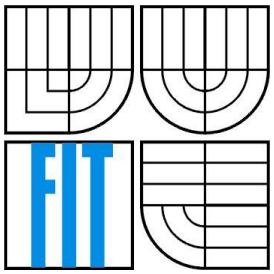
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE NIKOLAS DUBA
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SOFTWARE PRO PODPORU ONLINE HRY ENTROPIA UNIVERSE

SOFTWARE FOR A SUPPORT OF THE ENTROPIA UNIVERSE ONLINE GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

NIKOLAS DUBA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. GRULICH LUKÁŠ

BRNO 2008

Abstrakt

Táto práce slouží na pomoc hráčův počítačové online hry Entropia Universe. Pomocí software jsme schopný dosáhnout vyšší efektivitu v získávání surovin. Program požaduje jako vstup body (nálezy) zadávané jednotlivě nebo hromadně ze souboru. Po nastavení parametrů, filtrů a přesnosti jako výstup dostaneme grafické znázornění oblastí surovin nakreslenou na obrázek mapy, rozdělenou podle typu. Software umožňuje ukládání barevných schéma do souboru a jejich opětovné načítání. Body se ukládají do vnitrné databáze. Databázi je možné exportovat do souboru CSV anebo XML.

Klíčová slova

Mapa, oblast, nález, Entropia universe, hra, hrdiny, hráči, suroviny, svět, marching squares, visual studio, xaml, windows presentation foundation

Abstract

This software is for helping players of a Massive Multiplayer Online Role Playing Game: Entropia Universe. With this software we can achieve better results in finding resources during mining. Program as input needs points entered separately point by point or massively imported from a file. After properties setup program will visualize the resource area map. Every type has a different color. It allows saving color scheme settings into file and loading them when needed. Points are saved into database and can be exported into CSV or XML file.

Keywords

Map, area, finds, Entropia universe, game, role, massive multiplayer, resources, online, marching squares, visual studio, xaml, windows presentation foundation

Citace

Duba Nikolas: Software pro podporu online hry Entropia Universe. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Software pro podporu online hry Entropia Universe

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Lukáša Grulicha
Další informace jsem získal z internetových zdrojů, odborné literatury a z publikací.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Nikolas Duba
12.5.2008

Poděkování

Rád bych poděkoval Ing. Lukášovi Grulichovi, vedoucímu mé bakalářské práce, za připomínky a čas, který věnoval mé práci. Dále bych rád poděkoval všem těm, kdo mě učili a pomohli mi získat potřebné znalosti, abych dokázal tuto práci vůbec vytvořit. Také bych chtěl poděkovat svým rodičům, za jejich všestrannou podporu při studiu, bez které bych jistě tyto řádky vůbec nepsal. V neposlední řadě bych chtěl poděkovat všem těm, kdož si tuto práci přečetli, a tím mi ji pomohli udělat takovou, jaká teď je.

Obsah

Obsah 1
Úvod 3
1 O hre Entropia Universe 4
1.1 Popis herného žánru 4
1.2 Ekonomický systém 4
1.3 Dej 4
1.4 Svet Entropie 5
1.5 Možnosti v hre a miesto prepojenia s prácou 5
1.5.1 Profesie 5
1.5.2 Ťaženie surovín 6
1.5.3 Úloha práce v hre 7
2 Použité technológie 8
2.1 Vývojové prostredie 8
2.1.1 LINQ 9
2.1.2 Multi-Targeting 9
2.1.3 IntelliSense 9
2.2 Windows Presentation Foundation 10
2.2.1 Grafika 10
2.2.2 Rendering 10
2.2.3 XAML 10
3 Marching Squares 11
3.1 Pochodujúce kocky a jeho varianty 11
3.2 Pochodujúce štvorce (MS) 11
3.2.2 Vlastnosti Iso-ohraničenia 14
3.2.3 2D nejednoznačnosť 17
3.2.4 Asymptotický rozhodovač v 2D 18
4 Návrh a Implementácia 20
4.1 Návrh 20
4.2 Vzhľad aplikácie 20
4.3 Hlavné okno pomocou Visual Studio 2008 21
4.4 Uloženie dát 22
4.4.1 Transformácia súradníc 22
4.4.2 Postup kalibrácie mapy 23
4.4.3 Uloženie typov objektov 24

4.5 Implementácia Marching Squares 24
4.5.1 Zobrazenie marching squares vo WPF formulári 27
5 Záver 28
Literatúra 29

Úvod

Táto práca bola vytvorená účelom pomáhať hráčom on-line hry Entropia Universe typu MMORPG. Software pomáha hráčom jednoduchšie získavať žiadané suroviny v hre a to tak, že po zadaní potrebných informácií (nálezy) generuje polygóny na obrázku mapy pomocou, ktorých vidíme presne ohraničené oblasti. Oblasti majú rôzne typy, ktoré sa dajú rozlíšiť podľa ich farby. V týchto oblastiach môžeme nájsť požadovaný typ suroviny.

Prvá kapitola nás oboznámi s herným svetom. Nájdeme tam popis herného žánru, vysvetlenie ekonomického systému a samotný dej hry. V tejto kapitole môžeme získať potrebné informácie na pochopenie vytvoreného programu a popis ako sa dajú získať potrebné informácie na použitie softwaru.

Druhá kapitola nás oboznámi presnejšie s použitým programovacím jazykom, s vývojovým prostredím Visual Studio 2008 a s technológiami, ktoré boli použité v projekte.

Tretia kapitola sa zameriava hlavne na matematický popis použitého algoritmu Marching squares, ktorý je jadrom vytvárania oblastí na mape. Ďalej sa tu nachádzajú teoretické východiská riešeného problému.

Vo štvrtej kapitole nájdeme podrobnejší návrh aplikácie a postup spracovaní riešenej problematiky.

1 O hre Entropia Universe

1.1 Popis herného žánru

Entropia Universe (ďalej len Entropia) sa pomaly stáva ďalšou generáciou interaktívnej zábavy. Je to hra typu MMORPG, čo je skratka od Massive Multiplayer Online Role Playing Game. Tento pojem sa skladá z dvoch častí. RPG je v skutočnosti veľmi starý žáner – človek si vytvorí svoju postavu, ktorá bude reprezentovať dotyčnú osobu v hre. Každá postava začína s rovnakými vlastnosťami, ktoré potom rozvíja podľa vlastného výberu . Dôležité je že má naozaj veľký výber, lebo nad každým počiatočným výberom visí veľmi košatý strom rozhodnutí ako rozvíjať postavu ďalej. MMO, teda masívny on-linový multiplayer je nová prevratná zložka tohto žánru. Hráči boli už znudený, keď museli stále hrať proti počítačovému protivníkovi, ktorému doteraz chýba zdravý rozum (umelá inteligencia je ešte na nízkej úrovni). Pri MMORPG obvykle spolupracujete so skupinkou ľudí, aby ste dosiahli spoločný cieľ čo môže byť napríklad aj porazenie inej skupinky ľudí alebo príšery. Práve toto je tá nová prevratná vec – on-line spolupráca a komunikácia (MMORPG hry mávajú niekedy až veľmi sofistikovane vyriešenú komunikáciu) a veľmi schopný protivník, lebo je to živý človek/ľudia a nie počítač.

1.2 Ekonomický systém

Entropia má skutočný ekonomický systém, ktorý dovoľuje vám ako užívateľovi, aby ste menili skutočný život a vaše peniaze do PED (projekt entropia dollars) a keď sa tak rozhodnete, späť do reálnych peňazí. Pod skutočným ekonomickým systémom je myslené, že ekonómia hry je spojená s ekonómiou v reálnom živote a to tak, že sa stanovil kurz na prevod peňazí do hry. Hra Entropia je bezplatná nepotrebujete žiadne mesačné paušály alebo podobné poplatky. Celý prevodný systém bol vymyslený k tomu aby ste mohli dobehnúť ľudí, ktorý to už hrajú roky alebo aby ste si získali základné vybavenie k hre a v neposlednom rade k získaní času. Keďže v tejto hre sa jedná o reálne peniaze Švédska vláda previedla vyšetrovanie proti hre a presvedčili sa, či sa nejedná o hazardnú hru. Všetko čo je potrebné k hraniu je software, ktorý je distribuovaný rôznymi cestami. Software sa dá stiahnuť z oficiálnej stránky alebo je dostupný v rôznych počítačových časopisoch.

1.3 Dej

Sme na konci 21. storočia. Na Zemi sa zvyšuje populácia s veľkým tempom a blíži sa k tomu, že vyťaží všetky prírodné zdroje. Ľudstvo vidí už len jedno východisko, že sa bude rozširovať ďalej do vesmíru. Začali s výstavbou medzinárodnej vesmírnej stanice na orbitu Zeme. Za týmto pokračoval

rýchly rozvoj technologických vymožeností v oblastiach ako sú umelá inteligencia, kybernetika a cestovanie vo vesmíre. Takto vznikla vízia ktorá sa nazývala Odysseus Project. Tento projekt mal nájsť planétu vhodnú na život. Bolo vyslaných 7 vesmírnych lodí, ktoré riadili roboti. Jedna z týchto lodí našla vhodnú planétu na obývanie a poslala späť signál na Zem. Keď tam ľudia dorazili, ocitli sa pod paľbou. Do robotov sa dostal vírus, ktorý spôsobil to, že roboti zaútočili na ľudí. Boj skončil tým, že robotov vyhnali do susedného solárneho systému. Od tej doby roboti neustále útočia na túto planétu, ktorú nazvali Calypso. A na tomto mieste sa začína hra.

1.4 Svet Entropie

Hneď ako vstúpite do hry, budete si môcť vytvoriť svoju vlastnú trojrozmernú osobu. Budete sa môcť presvedčiť, že je to veľmi jednoduché. Entropia vám dá možnosť zažiť život vnútri v obrovskej virtuálnej realite. Táto postava vás bude prezentovať pri sociálnych interakcií s ľuďmi po celom svete tak isto ako aj pri využívaní on-line služieb neuveriteľného virtuálneho prostredia. Hra je robená k tomu, aby vám pomohla plniť všetky vaše sny či fantázie a dobrodružstva vnútri sveta science-fiction. Svet Calypso je pod stálym vývojom a na začiatku zahŕňa dva obrovské kontinenty s veľkými mestami, kde aj vy začnete svoj virtuálny život. Mestá vám ponúknu virtuálne ihrisko pre sociálne interakcie s ďalšími hráčmi v prospievajúcich sociálnych spoločenstvách. Svet obsahuje tiež rôzne formy herných inštitúcií, skutočné on-line služby a rôznorodosť virtuálnej zábavy. Spoločne so všetkými on-line užívateľmi, máte možnosť si zobrať aktívnu rolu vo vytvorení novej civilizácie. Môžete preskúmať nové kontinenty a pritom robiť rôzne činnosti, ktoré vám prinesú ďalšie a ďalšie možnosti vo svete. Musíte sa naučiť využívať všetky zdroje s ktorými disponujete , veľkú škálu rastúcich schopností (skill), kolektívnej práce a vybavenie k tomu, aby ste znovu získali stratený raj.

Entropia Universe je registrovaná ochranná známka MindArk AB. MindArk je spoločnosť zodpovedná za rozvoj Projektu Entropia Universe.

1.5 Možnosti v hre a miesto prepojenia s prácou

1.5.1 Profesie

Na to aby sme vedeli hru hrať, potrebujeme na začiatku hry vybrať smer (profesiu), s pomocou ktorej budeme získavať naše financie. Keďže skoro na všetky činnosti v hre potrebujeme investovať najprv par PEDov, musíme si dávať pozor aby sme nebankrotovali, čo je v prípade bez rozumného hrania veľmi jednoduché. Na začiatku hry, má každý postavu s rovnakými vlastnosťami. Môžeme sa rozhodnúť ktorý smer si vyberieme. V hre sú tri hlavné profesie a to lov, výroba vecí a ťažba surovín.

V prípade, že si vyberieme lov, naše financie získavame zabíjaním rôznych typov príšer v hre. Na to aby sme vedeli zabiť príšeru, potrebujeme náboje, čo je časť investovania. Okrem míňania nábojov sa opotrebovávajú aj používané veci. Financie získame späť vykorisťovaním príšery.

V prípade, že si vyberieme výrobu vecí sa ocitneme pri úplne iných podmienkach. Výrobca musí pozorovať trh v Entropii a vyrábať takú vec, ktorú potrebujú hráči do ostatných profesií najviac. Na výrobu vecí potrebujeme suroviny, ktoré zaobstará tretia profesia, teda zákopník. K surovinám sa výrobca dostane pomocou kontaktov v hre alebo cez centrálnu aukciu. Každá vec má svoj plán na výrobu, kde máme napísané všetko čo k nej potrebujeme a jej mieru úspechu.

Poslednú profesiu rozoberieme v ďalšej kapitole. Tieto profesie hrajú v hre kľúčovú rolu, lebo sú podstatou hry a bez nich by sme nevedeli ako získať späť naše stratené PEDy. Profesie závisia na sebe. To znamená, že jeden potrebuje druhého. Je to ako v reálnom živote.

1.5.2 Ťaženie surovín

Ako sme to už písali, táto profesia je veľmi dôležitá, lebo okrem toho že to vie uživiť jedného hráča teda zákopníka je veľmi dôležitá aj pre výrobcu vecí, ktorému dodáva suroviny k výrobe.

Ťažobný priemysel sa skladá z troch stupňov:

1) Musíme nájsť nález v zemi (deposit).
2) Vytiahnutie zo zeme do vášho inventára.
3) Čistenie, aby sa dalo použiť na výrobu alebo predaj.

Na to aby sme mohli začať ťažiť potrebujeme zohnať prístroje na ťaženie, ktoré sa dajú jednoducho kúpiť. Tieto prístroje sú:

1) Detctonator - detektor
2) Extractor – vyťahovač
3) Refiner – čistič
4) Bomb, probe – seizmická bomba alebo sonda podľa typu hľadanej suroviny.

Suroviny delíme do dvoch kategórií a to na obohatené materiály (energy materials) a na kamene (ore). Podľa toho o ktorý typ sa jedná sú rozličné aj prístroje. V prípade, že sme si zohnali potrebné vybavenie na ťaženie, potom sa vydáme do sveta, nájsť nejaké suroviny. Suroviny sú rozdelené do veľkých oblastí, ktoré majú rôzne veľkosti. V hre nevieme zistiť, aká surovina sa v danej oblasti nachádza.

Priebeh ťaženia spočíva v tom, že si musíme nasadiť detektor a náhodne skúšať nájsť surovinu vo svete. To, že niečo nájdeme vôbec nie je zaručené. V prípade nálezu nám začne blikať detektor a bude ukazovať smer a miesto kde sme danú surovinu našli. Na mieste sa objaví tyč (claim rod) na označenie miesta nálezu. Nasadíme vyťahovač a vytiahneme nájdenú surovinu do nášho inventára. Keď sme vytiahli všetko môžeme pokračovať v hľadaní. Keď sme si minuli všetky bomby, tak sa môžeme vrátiť do mesta, kde z nájdených surovín niečo vyrobíme alebo ich predáme.

1.5.3 Úloha práce v hre

Táto hra má viac ako 700 tisíc registrovaných užívateľov. Veľké percento hráčov volí práve byť zákopníkom v hre. Hráči pre väčší úspech zaznamenávajú miesta nálezov jednotlivých surovín. Je všeobecne známe, že oblasti týchto surovín sa nemenia a zápis nálezov na kúsok papiera alebo do textových súborov po jednom je neefektívne a neprehľadné. Práve tento software by mal pomôcť hráčom aby svoje nálezy zadávali do programu jednoducho a pohodlne a ako výsledok dostanú mapu, na ktorej budú prehľadne vyznačené oblasti. Je to aj preto užitočné, lebo hráč presne vie kde má hľadať danú surovinu. Software pomôže znížiť náklady na hľadanie surovín.

Obrázok 1.1 – Svet

Obrázok 1.2 - Prístroje

Obrázok 1.3 – Nález

Obrázok 1.4 - Ťaženie

2 Použité technológie

2.1 Vývojové prostredie

K vytvoreniu projektu som použil úplne nové vývojové prostredie Visual Studio 2008. Je to nová generácia vývojárskych nástrojov od spoločnosti Microsoft. Samotné Visual Studio obsahuje veľa zásadných noviniek, z ktorých väčšia časť sa týka tvorby webových stránok, ale okrem toho prináša veľmi veľa užitočných vylepšení, s pomocou ktorých vieme naprogramovať práve náš software. Tento nástroj je vhodný pre začiatočníka, ale aj pre pokročilých. Podporuje moderné metodiky a princípy riadenia životného cyklu vývoja softwarových aplikácií. Poskytuje pokročilé vývojové nástroje, ladiace funkcie, funkcie pre prácu s databázou a vynaliezavé novinky pre rýchlu tvorbu špičkových aplikácií. Visual Studio aktívne spolupracuje s vývojovo-exekučnou platformou .NET Framework verzia 3.5, ktorý predstavuje ďalší stupeň verzie 3.0. Dot NET Framework 3.5 prišiel s nástupom Windows Vista ponúkajúc nové API pre budovanie riadených aplikácii. Zahŕňa tiež štyri znova zaradených aplikačných programových rozhraní:

- Windows Presentation Foundation (WPF): grafický subsystém s podporou multimédií a animácií
- Windows Communication Foundation (WCF): subsystém pre stavbu servisne orientovaných webových aplikácií podľa princípu SOA.
- Windows Workflow Foundation (WF): subsystém pre modelovaní toku procesov.
- Windows CardSpace: subsystém pre správu digitálnych entít.

Medzi neposledné vlastnosti patrí funkcia, ktorá umožňuje aplikáciami WinForms A WPF využívať služby ASP.NET aplikácií pre zdieľanie užívateľských dát.

Zhrnutie ďalších vlastností nástroja:

- Automatic Properties, object initializer and Collection Initializers – zjednoduší vytváranie vlastností, kolekcií a ďalších objektov.
- Extension Methods – umožňuje rozšíriť verejné rozhranie typov bez prístupu ich kódu.
- Lambda Expressions – je ďalším stupňom vývoja anonymných metód známych z .NET 2.0
- Anonymous Types – Vlastnosť známa z funkčných jazykov, ktorú sa dá použiť len pre lokálne premenné. Používa sa kľúčové slovo VAR a typ výrazu je takto odvodená z kontextu.
- Query syntax – ponúka stručnú deklaratívnu syntax pre vyjadrenie dotazov. Využíva štandardné operátory jazyka LINQ.

2.1.1 LINQ

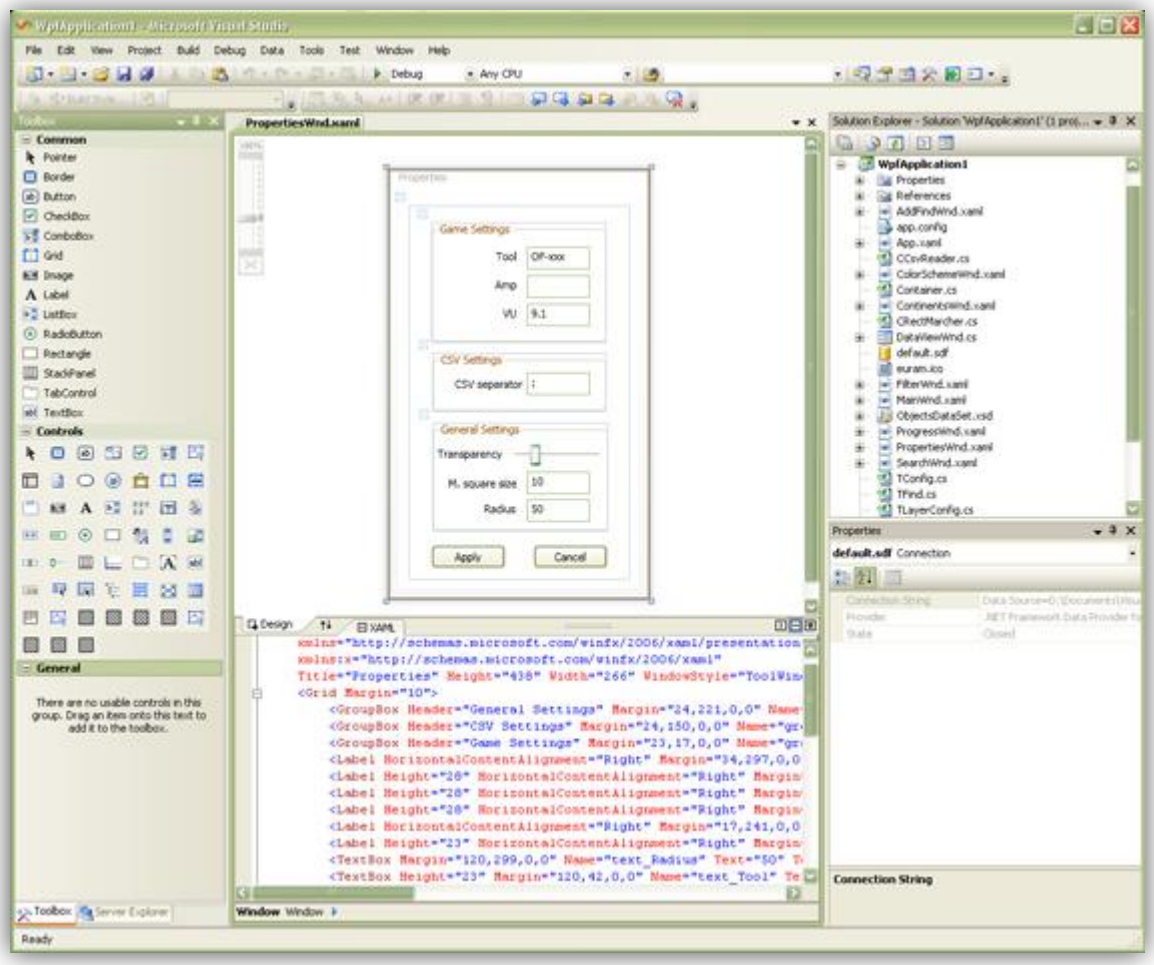
Nové kompilátory jazykov Visual Basic a C# prináša vylepšenie obidvoch jazykov. Objavujú sa prvky funkcionálneho programovania, ktoré umožňujú písať čistý a pôsobivejší kód. LINQ (Microsoft Language Integrated Query) je jazyk, ktorý zjednoduší dotazovanie a prácu s dátami.

2.1.2 Multi-Targeting

Umožňuje stavať aplikácie zamerané na rôzne verzie .NET frameworku. Konkrétne .NET FW 2.0, .NET FW 3.0 a nový .NET FW 3.5.

2.1.3 IntelliSense

Technológia Intellisense je funkcia zobrazení kódov, ktorá poskytuje automatické doplňovanie príkazov a taktiež kontextovo závislé zoznamy obsahujúce kód a skriptovacie prvky. Je to technológia, ktorá zvyšuje rýchlosť a efektívnosť programovania.



Obrázok 2.1 – Visual Studio 2008

2.2 Windows Presentation Foundation

Je podsystém .NET Framework, ktorý v sebe ukrýva jednotný prístup k UI dokumentom a médiám. Využíva sa to v aplikáciách na dosiahnutie bohatého grafického rozhrania.

2.2.1 Grafika

WPF je zameraná na graficky bohaté aplikácie a môže teda využívať možnosti vektorovej grafiky, animácií, multimédií, efektov a samozrejme 3D grafiky. Pri zmene veľkosti okna sú grafické elementy bezstratové vďaka technológii založenej na DPI, farebných gardientov a používania geometrických tvarov. Animácie sú založené časovo alebo na frameoch. WPF jednoducho ovláda multimediálne prvky ako su: mp3, wma, avi, mpeg a wmv. Komponent zvláda tiež efekty priehľadnosti, zrkadlení a stien.

2.2.2 Rendering

Možno sa zdá že takáto aplikácia môže veľmi zaťažovať procesor, ale nie je to tak. WPF je založené na DirectX. To znamená, že grafika je renderovaná na grafickej karte a nie na CPU. Grafické jadro bolo programované tak aby podporovalo verzie od DirectX 7 až 9. Vertex a Pixel Shader musí byť vo verzii 2.0 inak nastane situácia, že naša aplikácia bude renderovaná čiastočne na CPU a čiastočne na GPU. Najhorší variant je, keď celá aplikácia je renderovaná na CPU. V tomto prípade sa stane veľmi nepríjemné spomalenie počítača.

2.2.3 XAML

XAML prináša do programovania úplne novú vec. Ide o deklaratívne programovanie s pomocou, ktorého vieme oddeliť aplikačnú logiku od designu. V jednom súbore je XAML kód a v druhom súbore je program písaný v niektorom z .NET jazykov. Všetko robíme prostredníctvom značkovacieho jazyka XAML. Jeho hlavné výhody sú čistota, jednoduchosť a rýchlosť písania kódu. Jazyk XAML je niečo také ako keby sme programovali v HTML. Keď skompilujeme náš program, tak XAML kód sa prevedie do zdrojového kódu a následne je preložená do binárnej podoby. XAML nemusí slúžiť len ako definícia vzhľadu, ale môže to byť aj definícia systémových zdrojov alebo ako celá aplikácia.

3 Marching Squares

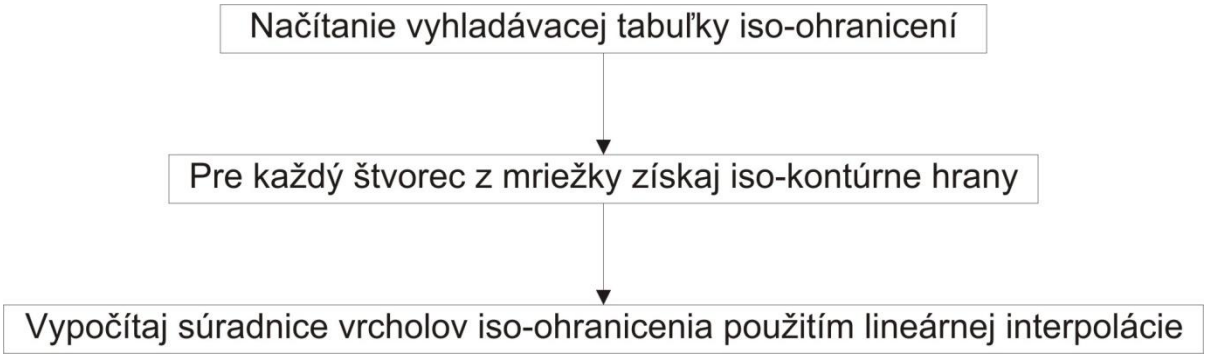
3.1 Pochodujúce kocky a jeho varianty

Algoritmus Marching Cubes - Pochodujúce kocky (MC) - je založený na dvoch myšlienkach. Po prvé, iso-povrch môže byt vytvorený po častiach vnútri každej kocky, bez použitia ostatných kociek mriežky. Po druhé, kombinatorická štruktúra každej záplaty iso-povrchu v kocke mriežky môže byť získaná z vyhľadávacej tabuľky. Hlavná operácia slúži na získavanie tejto štruktúry z vyhľadávacej tabuľky. Vedľajšia operácia je algoritmus, ktorý beží v čase úmernom počtu kociek tvoriacich mriežku.

3.2 Pochodujúce štvorce (MS)

Zoberme si dvojrozmernú mriežku vzorkujúcu skalárnu plochu. Jednoduchý príklad je obrázok v odtieňoch sivej. Vzorky sú odobrané zo stredných pixlov. Skalárnu hodnotu získame z výpočtu sivosti pixlov.

Všimnite si, že obrázok v odtieňoch sivej má dve prirodzené mriežky. Mriežku definujúcu hranice pixlov a mriežku obsahujúce stredy pixlov. Tieto dve mriežky sú navzájom "duálne". Na vykonanie iso-ohraničenia potrebujeme mriežku, ktorej body sú stredy pixlov.
Vstupov MS algoritmu je iso-hodnota a množina skalárnych hodnôt vo vrcholoch dvojrozmernej pravidelnej mriežky. Algoritmus



Obrázok 3.1

má tri kroky. (Pozri obrázok 3.1). Načítanie vyhľadávacej tabuľky iso-ohranicení z vopred pripraveného súboru. Pre každý štvorec získaj z vyhľadávacej tabuľky množinu hrán, reprezentujúce kombinatorickú štruktúru iso-ohraničenia. Koncové body týchto hrán tvoria vrcholy iso-ohraničenia. Priradíme geometrické pozície k vrcholom iso-ohraničenia podľa skalárnych hodnôt v koncových bodoch hrán štvorca.

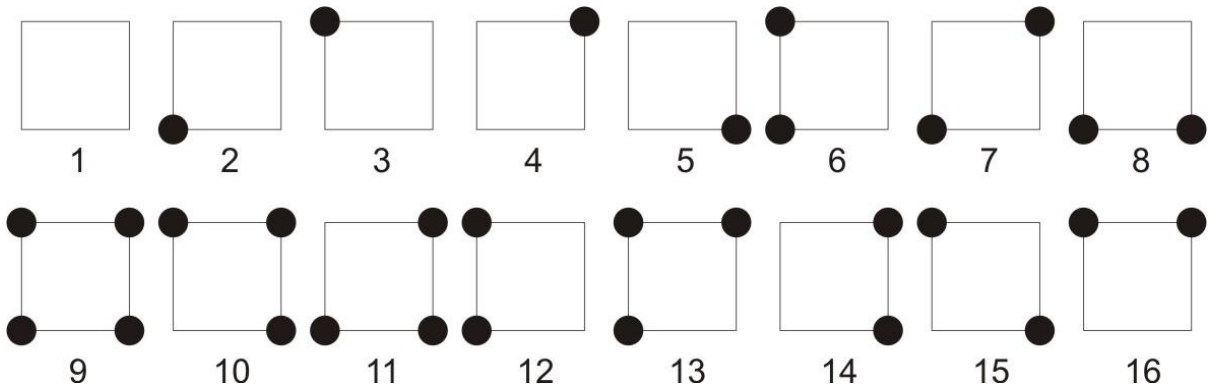
Každý vrchol mriežky má skalárnu hodnotu menšiu, rovnú alebo väčšiu ako iso-hodnota. Označíme vrchol ako 'kladný' '+', ak je jeho hodnota väčšia alebo rovná iso-hodnote. 'Záporný' '-', ak je jeho hodnota menšia ako iso-hodnota. Kladný vrchol nazveme rýdzo kladným, ak sa jeho hodnota nerovná iso-hodnote. Keďže skalárna hodnota negatívneho vrcholu sa nikdy nerovná iso-hodnote, nie je dôvoď definovať podobný 'rýdzo záporný' výraz.

Hrany mriežky, ktorých oba koncové body sú 'kladné' sa nazývajú 'kladné hrany'. Hrany mriežky, ktorých oba koncové body sú negatívne sa nazývajú 'záporné hrany'. Hrany s jedným vrcholom 'kladným' a druhým 'záporným' sa nazývajú '(+-)-hrany'.

Keďže každý vrchol je buď kladný alebo záporný a štvorec má štyri vrcholy, máme $2^4 = 16$ rôznych konfigurácií označení vrcholov štvorca.

Kombinatorická štruktúra iso-ohraničenia vnútri každého štvorca je určená z konfigurácií označení vrcholov štvorca. Aby sa oddelili kladné vrcholy od záporných, iso-ohraničenie musí pretínať nijakú hranu štvorca, ktorá má jeden kladný a jeden záporný koncový bod. Iso-ohraničenie ktorá pretína minimálny počet hrán mriežky nebude pretínať žiadnu hranu štvorca, ktorej koncové body sú oba rýdzo kladné alebo rýdzo záporné.

Pre každú konfiguráciu štvorca C, nech je množina (+-)-hrán. Všimnite si, že veľkosť $E_C^{+/-}$ je buď nula, dva alebo štyri. Spárte hrany z $E_C^{+/-}$. Každý takýto pár predstavuje hranu iso-ohraničenia s koncovými bodmi na dvoch členoch páru. Obrázok 3.2 obsahuje 16 konfigurácií štvorcov a ich iso-ohraničení. Vyhľadávacia tabuľka iso-ohraničení, ktorá sa nazýva *Table*, obsahuje šestnásť konfigurácií. Každá konfigurácia, *Table*(C) je zoznam $E_C^{+/-}$. párov.



Obrázok 3.2

Na obrázku 3.2, sú zobrazené hrany iso-ohraničenia spájajúce stred každej hrany štvorca. Toto je iba ukážka. Geometrické pozície vrcholov iso-ohraničenia nie sú definované vyhľadávacou tabuľkou.

Vyhľadávacia tabuľka iso-ohraničenia je skonštruovaná na jednotkovom štvorci s vrcholmi $(0,0), (0,1), (1,0), (1,1)$. Na skonštruovanie hrán iso-ohraničenia v štvorci mriežky (i,j), musíme priradiť páry hrán jednotkového štvorca na páry (i,j) hrán. Každý vrchol $v = (vx, vy)$ jednotkového štvorca sa priradí na $v + (i,j) = (vx, vy) + (i,j) = (vx + i, vy + j)$. Každá hrana e jednotkového štvorca s koncovými bodmi (v,v') sa priradí hrane $e + (i,j) = (v + (i,j), v' + (i,j))$. Nakoniec sa každý pár hrán $(e1, e2)$ priradí na $(e1 + (i,j), e2 + (i,j))$.

Ak štvorec mriežky (i,j) má konfiguráciu *C*, potom hranový pár $(e_1, e_2) \in Tabuľka|C|$ sa zhoduje s hranovým párom. Koncové body hrán iso-ohraničenia sú vrcholy iso-ohraničenia. Na priradenie každej hrany iso-ohraničenia časti geometrickej čiary, používame lineárnu interpoláciu na umiestnenie vrcholov iso-ohraničenia. Každý vrchol iso-ohraničenia leží na hrane mriežky [p,q]. Ak sp a sq sú skalárne hodnoty v p a q a σ je iso-hodnota, potom priradíme v na $\alpha \cdot p + (1-\alpha) \cdot q$, kde $\alpha = (sp = \sigma)/(sq - sp)$. Všimnite si, že keď p a q majú rôzne znamienko, skalárny *sp* sa nerovná *sq* a menovateľ $(sq - sp)$ je nenulový.

MS algoritmus je uvedený v Algoritme 3.2. Funkcia LinearInterpolation2D, ktorý MS algoritmus volá:

Vstup :$Body\ p, q \in R^2, skalárne\ hodnoty\ s_p, s_q, a\ isohodnota\ \sigma$

Požiadavky :$s_p \neq s_q\ a\ zároveň\ s_p \leq \sigma \leq s_q$ alebo $s_p \geq \sigma \geq s_q$.

Výstup :$Bod\ r\ leží\ na\ [p, q]$

LinearInterpolation2D(p, s_p, q, s_q, σ)

1 $\alpha \leftarrow \frac{s_q - \sigma}{s_p - s_q}$;

2 $r_x \leftarrow \alpha p_x + (1-\alpha) q_x$;

3 $r_y \leftarrow \alpha p_y + (1-\alpha) q_y$;

4 return (r);

Algoritmus 3.1: LinearInterpolation2D

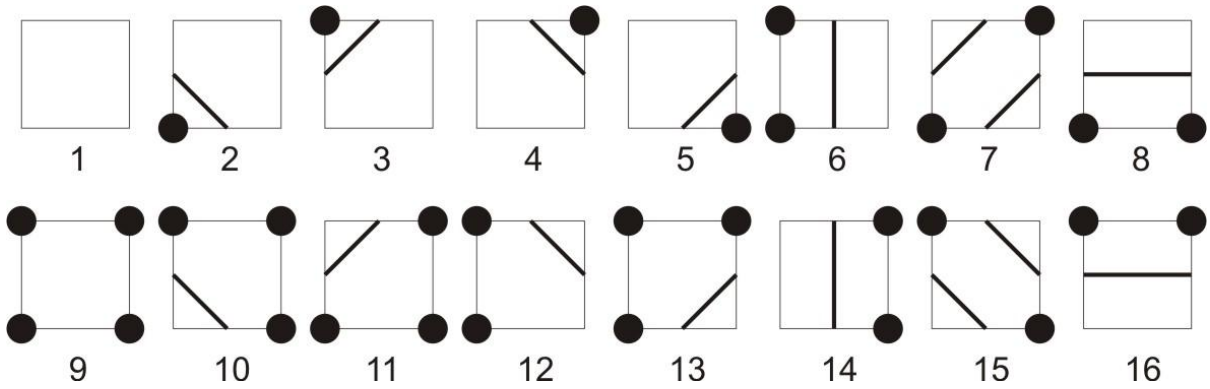
3.2.2 Vlastnosti Iso-ohraničenia

Aby sme správne prediskutovali výstup vytvorený MS algoritmom, musíme rozlišovať medzi dvoma prípadmi v závislosti na iso-hodnote. V prvom prípade sa iso-hodnota nerovná skalárnej hodnote žiadneho vrcholu mriežky a MS algoritmus vytvorí po častiach lineárnu, orientovanú 1- obťah s okrajom. Hranica 1- obťah leží na okraji mriežky. V druhom prípade je iso-hodnota rovná skalárnej hodnote jedného alebo viacerých

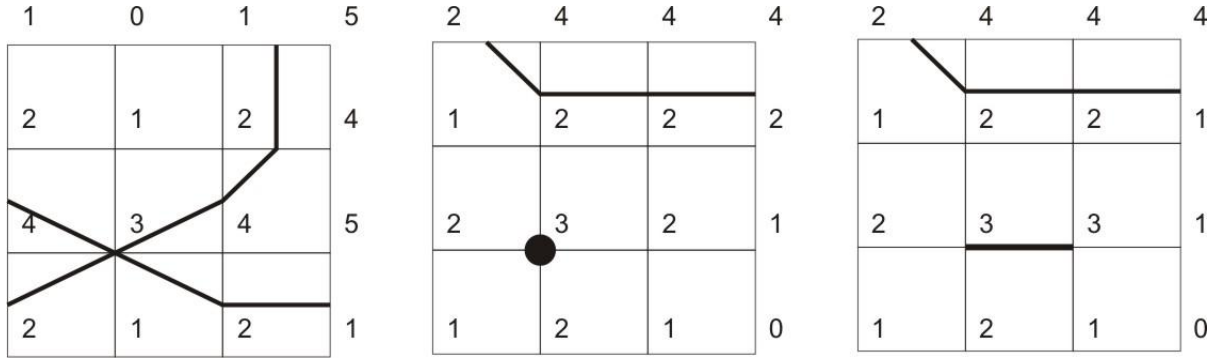
Vstup: F je 2D pole skalárnych hodnôt.
Coord je 2D pole (x, y) koordinát.
σ je isohodnota.
Výsledok: Sada Γ isokontúrových lineárnych segmentov .

1 Načítaj MS vyhľadávaciu tabuľku do $Table$;
/* *Priraď ‚+' alebo ‚-' značky pre všetky vrcholy (vertex)* */
2 **foreach** grid vertex (i,j) **do**
3 | **if** $F[i,j] < \sigma$ **then** $Sign[i,j] \leftarrow' -'$;
4 | **else** $Sign[i,j] \leftarrow' +'$; /* $F[i,j] \geq \sigma$ */
5 **end**
6 $S \leftarrow \emptyset$;
/* *Pre všetky štvorce mriežky získaj iso-kontúrne hrany* */
7 **foreach** grid square (i,j) **do**
/* *Vrcholy štvorcových mriežok sú (i,j),(i,j+1),(i+1,j),(i+1,j+1)* */
8 | $C \leftarrow (Sign[i,j], Sign[i,j+1], Sign[i+1,j], Sign[i+1,j+1])$;
9 | **foreach** edge pair $(e_1, e_2) \in Table[C]$ **do**
10 | | **insert** edge pair $(e_1 + (i,j), e_2 + (i,j))$ into S;
11 | **end**
12 **end**
/* *Vypočítaj koordináty iso-konturných vrcholov pomocou lineárnej interpolácii* */
13 **foreach** $(+-)$-grid edge e with endpoints (i_1, j_1) and (i_2, j_2) **do**
14 | $r_e \leftarrow$LinearInterpolation2D$(Coord[i_1, j_1], F[i_1, j_1], Coord[i_2, j_2], F[i_2, j_2], \sigma)$
15 **end**
/*Premeň S na sadu lineárnych segmentov*/
16 $\Gamma \leftarrow \emptyset$
17 **foreach** pair of edges $(e_1, e_2) \in S$ **do**
18 | $\Gamma \leftarrow \Gamma \cup \{r_{e_1}, r_{e_2}\}$;
19 **end**

Algoritmus 3.2: Marching Squares



Obrázok 3.3

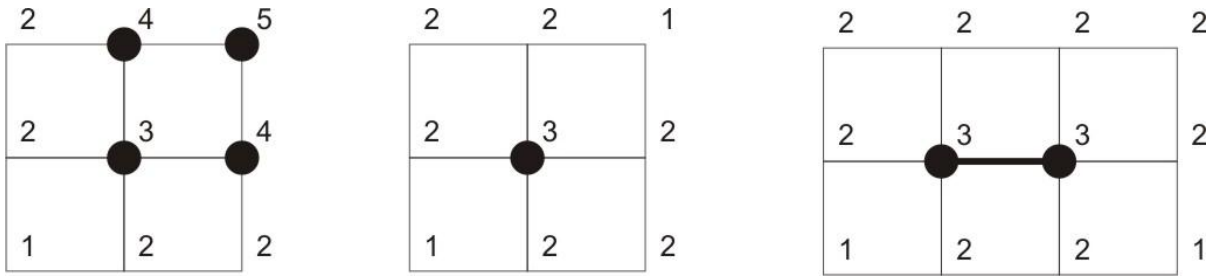


Obrázok 3.4

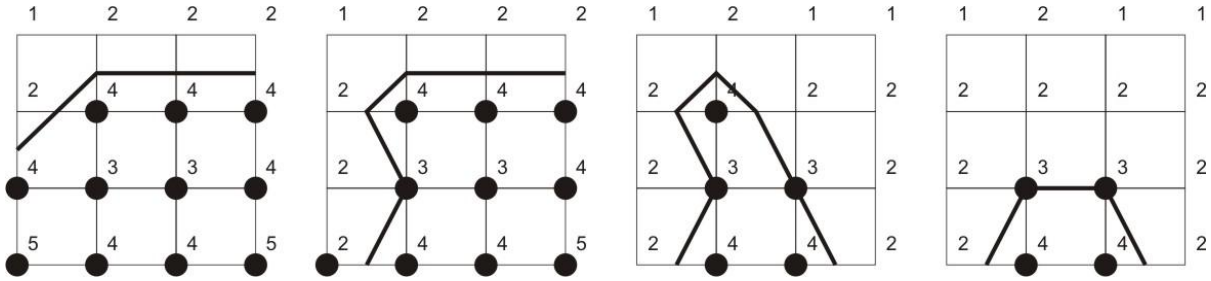
vrcholov mriežky. V tomto prípade môže MS algoritmus vytvoriť 1-obťah s okrajom alebo okraj nemusí ležať na kraji mriežky. Napr. MS algoritmus použitý na mriežku 3x3 v obrázku 3.4 vytvorí neobťahové iso-ohraničenie alebo iso-ohraničenie s okrajom mimo skalárnej mriežky. V prvej mriežke sa pretínajú štyri čiarové úseky iso-ohraničenia v jednom bode, v druhej mriežke je iso-ohraničenie samotný bod a v tretej mriežke leží hranica iso-ohraničenia v mriežke.

Tieto dva prípady sa tiež odlišujú v druhu čiarových úsekov tvorených algoritmom. Iso-ohraničenie vytvorené MS algoritmom je množina čiarových úsekov ktorých vrcholy ležia na hranách mriežky. Ak je iso-hodnota rôzna od skalárnych hodnôt každého vrcholu mriežky, potom majú všetky tieto čiarové úseky kladnú dĺžku. Ak sa iso-hodnota rovná skalárnej hodnote jedného alebo viacerých vrcholov, potom môže mať iso-ohraničenie hrany s nulovou dĺžkou. Napríklad MS algoritmus použitý na tri mriežky na obrázku 3.5 vytvorí iso-ohraničenie pre hodnotu 3 s hranami dĺžky nula.

V najľavejšom obrázku na obrázku 3.5, spodná ľavá mriežka má konfiguráciu 4 vytvárajúcu jedinú hranu iso-ohraničenia, ale oba koncové body tej hrany sa priradia na vrchol v strede mriežky. Na prostrednom obrázku vytvára každý štvorec mriežky hranu iso-ohraničenia, ale všetky štyri hrany majú nulovú dĺžku a zbalia sa do jedného bodu. V najpravejšom obrázku najľavejší a najpravejší štvorec



Obrázok 3.5: Priklady hrán dĺžky nula vytvorených MS algoritmom (iso-hodnota 3). Čierne vrcholy majú skalárnu hodnotu väčšiu alebo rovnú 3 (kladné vrcholy). Najľavejší obrázok obsahuje jednu hranu iso-ohranicenia dĺžky nula (z ľavého dolného štvorca mriežky), a prostredný a pravý obrázok obsahujú štyri hrany iso-ohranicenia dĺžky nula. Najpravejší obrázok tiež obsahuje dve nenulové hrany isoohranicenia (z prostredných štvorcov mriežky) ktoré sa priradia na rovnaké hrany mriežky.



Obrázok 3.6: Príklad hrany mriežky s oboma skalárnymi hodnotami koncových bodov rovných iso-hodnote (3) preťaté nula, raz a dva krát iso-ohranicením. Čierne vrcholy majú skalárnu hodnotu väčšiu alebo rovnú 3 (kladné vrcholy).

vytvárajú hrany iso-ohraničenia dĺžky nula. Zvyšné dva prostredné štvorce mriežky vytvárajú dve hrany iso-ohraničenia, ktoré sa priradia na rovnakú hranu mriežky.

MS algoritmus vráti konečnú množinu čiarových úsekov. Iso-ohraničenie je zjednotenie týchto čiarových úsekov. Vrcholy iso-ohraničenia sú koncové body čiarových úsekov.
Nasledovné vlastnosti platia pre všetky iso-ohraničenia vytvorené MS algoritmom:

1. Iso-ohraničenie je po častiach lineárne
2. Vrcholy iso-ohraničenia ležia na hranách mriežky
3. Iso-ohraničenie pretína každú hranu mriežky najviac jeden krát, okrem prípadu, keď sú obe skalárne hodnoty koncových bodov rovné iso-hodnote
4. Iso-ohraničenie nepretína žiadne hrany mriežky, ktorých oba koncové body majú skalárnu hodnotu väčšiu alebo oba menšiu ako iso-hodnota.
5. Iso-ohraničenie oddeľuje vrcholy mriežky so skalárnymi hodnotami väčšími alebo rovnými iso-hodnote od vrcholov mriežky so skalárnymi hodnotami menšími ako iso-hodnota. Úplne ich oddeľuje v prípade, že ich skalárna hodnota sa nerovná iso-hodnote.

Vlastnosti 3 a 4 znamenajú, že iso-ohraničenie pretína minimálny počet hrán mriežky. Ak majú oba konce hrany mriežky skalárnu hodnotu rovnú iso-hodnote, potom iso-ohraničenie môže pretínať hranu mriežky raz, dvakrát alebo nemusí vôbec. (Pozri obrázok 3.6)

Vlastnosť 5 znamená, že iso-ohraničenie pretína každú (+-) hranu mriežky. Avšak (+-) hrany mriežky môžu byt preťaté hranou iso-ohraničenia z nulovou dĺžkou ako v prostrednom a pravom obrázku v obrázku 3.5.

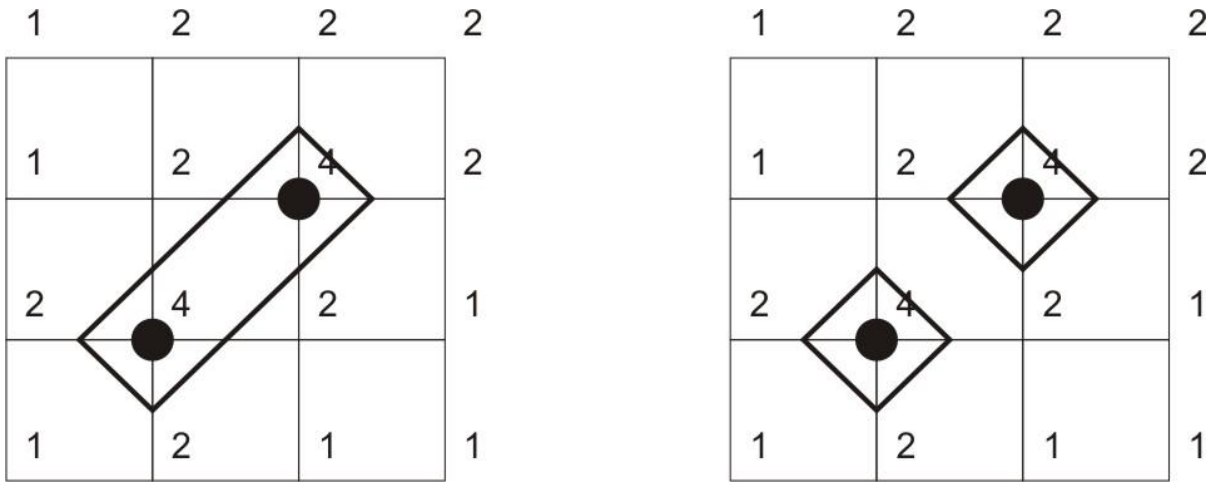
Nasledovné vlastnosti platia pre MS iso-ohraničenia, ktorých iso-hodnota sa nerovná skalárnej hodnote žiadneho vrcholu mriežky.

6. Iso-ohraničenie je po častiach lineárne 1-obťah s okrajom
7. Okraj iso-ohraničenia leží na okraji mriežky
8. Množina Γ neobsahuje žiadny čiarový úsek s nulovou dĺžkou. Zdvojené čiarové úseky a čiarové úseky v Γ tvoria "trianguláciu" iso-ohraničenia.

Triangulácia vo vlastnosti 8 jednoducho znamená, že čiarové úseky v Γ sa pretínajú v ich koncových bodoch. Iso-ohraničenie je jednorozmerné a neobsahuje trojuholníky.

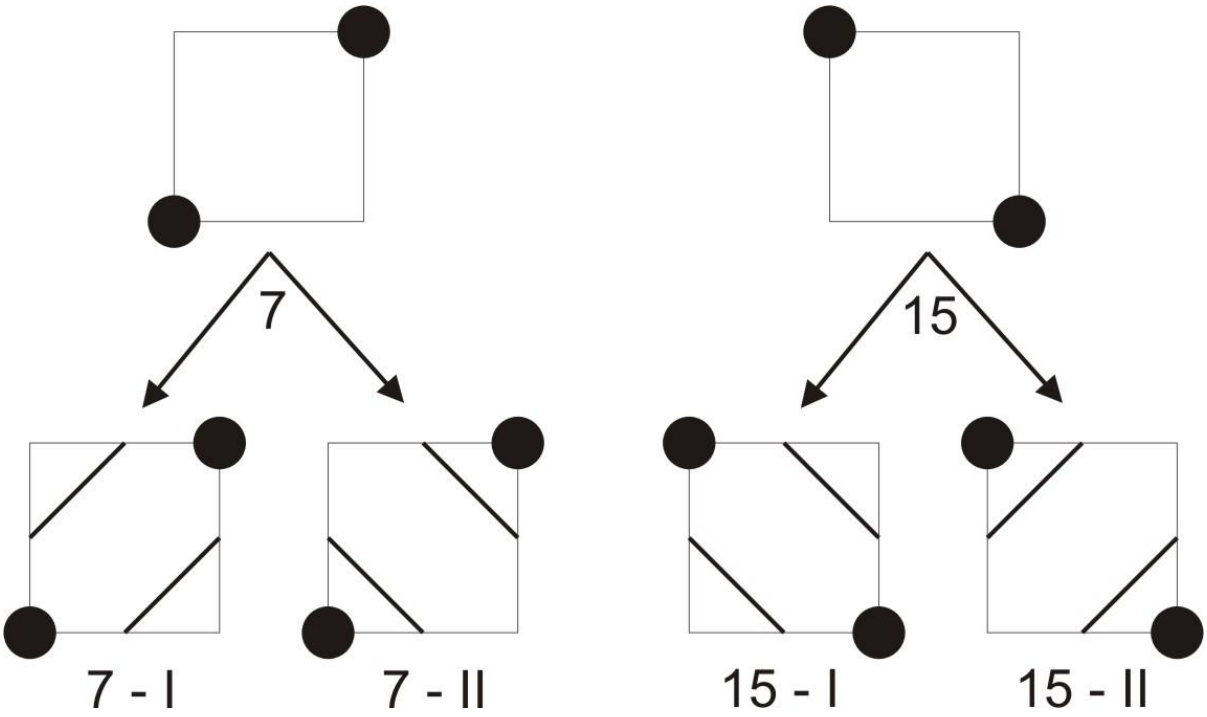
3.2.3 2D nejednoznačnosť

Množina $E_C^{+/-}$ je množina (+-) hrán štvorcov pre konfigurácie *C*. Kombinatorická štruktúra iso-povrchu záleží na porovnávaní prvkov z $E_C^{+/-}$. Ak má $E_C^{+/-}$ dva prvky, potom nie je výber. Ak má ale $E_C^{+/-}$



Obrázok 3.7: Typologicky rôzne iso-ohranicenia vytvorene použitím rôznych iso-ohraniceni pre viac významovú konfiguráciu v centrálnom štvorci mriežky.

štyri prvky, potom sú možné dve párenia a môžu byť vytvorené dve rôzne iso-ohraničenia v štvorci *C*. Dve zo šestnástich konfigurácií majú takéto prvky. Volajú sa nejednoznačné konfigurácie. Dve nejednoznačné konfigurácie sú ukázané na obrázku 3.8 spolu s dvoma kombinatoricky rôznymi iso-ohraničeniami pre každú nejednoznačnú konfiguráciu.



Obrázok 3.8

Vybranie rôzneho iso-ohraničenia pre rôzne konfigurácie zmení topológiu celého iso-ohraničenia. Napríklad, obrázok 3.7 ukazuje rovnakú skalárnu mriežku s dvoma typologicky odlišnými iso-ohraničeniami vytvorenými rôznymi riešeniami nejednoznačných konfigurácií. Prvé iso-ohraničenie má dva komponenty, zatiaľ čo druhé má iba jeden.

3.2.4 Asymptotický rozhodovač v 2D

MS algoritmus opravuje, fixuje, ukladá iso-ohraničenie pre každú konfiguráciu vo vyhľadávacej tabuľke a vždy použije toto iso-ohraničenie. Preto štvorce s rovnakou konfiguráciou vždy obsahujú kombinatoricky ekvivalentné iso-ohraničenia. Greg Nielsen a Bern Hamann [NH91] odporučili, aby bol každý štvorec s nejednoznačnou konfiguráciou vrcholov rozhodnutý použitím *bilineárnej* interpolácie.

Uvažujeme jednotkový štvorec s vrcholmi (0,0), (0,1), (1,0) a (1,1) so skalárnymi hodnotami $s_{0,0}, s_{0,1}, s_{1,0}$ a $s_{1,1}$ v tomto poradí. Body v jednotkovom štvorci sú (x,y) kde $0 \leq x \leq 1$ a $0 \leq y \leq 1$. Funkcia $f: R^2 \rightarrow R$ kde

$$f(x,y) = (1-x, x) \cdot \begin{pmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{pmatrix} \begin{pmatrix} 1-y \\ y \end{pmatrix}$$

Rovnica 3.1

Pre takmer všetky $c \in R$, úrovňová množina $\{(x,y): f(x,y) = c\}$ je tvorená dvomi hyperbolami. V nejednoznačnej konfigurácii pretínajú oba hyperboly jednotkový štvorec. Sú dve možné spôsoby ako. Každý z týchto spôsobov spája rôzny pár hrán štvorcov mriežky zodpovedajúce jednej z dvoch kombinatoricky rôznych po častiach lineárnych iso-ohraničení. Preto môže byť nejednoznačná konfigurácia rozhodnutá vytvorením bilineárného interpolantu a určením, ako pretnú tieto dve hyperboly štvorcovú mriežku.

Nie je nutné skutočne konštruovať hyperboly aby sme určili, ako pretnú jednotkový štvorec. Namiesto toho vieme určiť hodnotu ohraničenia α takú, že $B(s,t) = \alpha$ je pretínajúca horizontálne a vertikálne asymptoty. Ak je iso-hodnota väčšia ako α, potom hyperboly pretínajú jednotkový štvorec ako v prípade 1, inak sa jednotkové štvorce pretínajú ako v prípade 2.

Rovnica 3.1 sa rozšíri na:

$$f(x,y) = s_{0,0} + x(s_{1,0} - s_{0,0}) + y(s_{0,1} - s_{0,0}) + xy(s_{0,0} - s_{0,1} - s_{1,0} + s_{1,1}).$$

Rovnica 3.2

Asymptóty sú určené buď výberom *x* tak, aby koeficient *y* bol nulový alebo výberom *y* tak, aby koeficient *x* bol nulový.

$$\left\{(x,y): x = \frac{s_{0,0} - s_{0,1}}{s_{0,0} + s_{1,1} - s_{0,1} - s_{1,0}}\right\}$$
$$\left\{(x,y): y = \frac{s_{0,0} - s_{0,1}}{s_{0,0} + s_{1,1} - s_{0,1} - s_{1,0}}\right\}$$

Použitím týchto hodnôt spätne do $f(x,y)$ nám dá

$$\alpha = \frac{s_{0,0}s_{1,1} + s_{1,0}s_{0,1}}{s_{0,0} + s_{1,1} - s_{0,1} - s_{1,0}}.$$

Nejednoznačná konfigurácia je rozhodnutá porovnaním iso-hodnoty k tejto α.

4 Návrh a Implementácia

4.1 Návrh

Na začiatku je treba si ujasniť požiadavky, ktoré budeme klásť na funkcionalitu programu. Predovšetkým ide o katalogizovanie, hľadanie a editáciu záznamov o nálezoch nerastných surovín v Entropia Universe. Program bude samostatná aplikácia, nebude sa teda jednať o doplnok (plug-in / add-in) do hry samotnej. Po dohode s vedúcim práce bude používaným programovacím jazykom C#.

Program by mal umožňovať prehľadné a jednoduché zobrazenie herného sveta, k čomu úplne postačí 2D mapa sveta, ktorá je uložená ako rastrový obrázok. Je samozrejmé že mapu bude potrebné zväčšovať a posúvať ju. Na mape sa potom zobrazujú samotné nálezy, ktoré sú vo forme bodov a po spustení príkazu ku generovaní marching squares sa zobrazujú ako ohraničené priehľadné povrchy. Každý typ suroviny je pritom zobrazený v samostatnej vrstve, prevedené prednastavenou farbou. Ktoré vrstvy sa zobrazia a akou farbou sa vykreslia pritom užívateľ môže plne nastaviť.

Nálezy sa zadávajú buď ručne, alebo kliknutím na mapu a nastavením typu nálezu alebo pomocou importu tabuľky nálezov z .csv nebo .xml súboru, vygenerovaného inou utilitou. Program si pritom udržuje svoju globálnu databázu nálezov, ktorú zobrazuje na mape. Možnosť celú databázu importovať a exportovať do .csv a .xml je tiež samozrejmosťou. V záujmu kompatibility sú pritom použité rovnaké formáty súborov ako u cudzích utilít.

Databáze je v podstate len zoznam záznamov, neuvažujú sa nijaké vzťahy medzi nálezy a podobne, takže by postačilo uložiť záznamy jednoducho v ich „natívnej“ forme – xml súboru, avšak nálezy budeme neskôr chcieť prezerať, vyhľadávať v nich a podobne a sprievodcovia Visual Stúdia za nás urobia kus práce pri vytváraní editačného dialógu, práve keď budú záznamy v databázovej tabuľke.

Program si udržuje všetky svoje nastavenia a farebné schémy v jednoduchej sade .csv súborov, preto je pre tento účel úplne postačujúci. S csv súbormi sa dá veľmi jednoducho manipulovať na rozdiel od databázy. K editácii csv súboru nie je potrebná žiadna zo špecializovaných nástrojov.

4.2 Vzhľad aplikácie

Aplikácia sa skladá z hlavného okna, v ktorom dominuje zobrazenie mapy. Pod mapou je posuvník, ktorý ovláda mierku mapy. Maximálna hodnota zväčšení je desaťkrát, minimálna hodnota je závislá na relatívnej veľkosti okna a mapy a je určovaná tak, aby sa pri minimálnom zväčšení, mapa práve vošla do okna celá. Bol pritom kladený dôraz na užívateľský komfort aby sa mapu dalo posúvať ťahaním s myšou a zväčšovať otáčaním jeho kolieska. (Za poznamenanie stojí, že zväčšovanie mapy je kompletne implementovaná v xaml a nebol pre neho napísaný jediný riadok C#. Xaml totiž

odkrýva transformačné funkcie grafickej karty a každá vizuálna entita má vlastnosť transformácie. Výhodou je jednoduchý a rýchly vývoj podobných aplikácií, nevýhodou je mierna obmedzenosť. Mapa sa zväčšuje pomocou implicitného bilineárného filtrovania grafickej karty. Xaml je však uzavretý a neumožňuje priamo použiť *pixel shader*, ktorý by implementoval napríklad *bikubické* filtrovanie. Ale je to pomerne nový štandard, ktorý sa neustále vyvíja).

Pokračujeme popisom hlavného okna aplikácie. V ľavom spodnom rohu je pozícia kurzoru myši na mape, jednak v pixeloch (počítali sme aj s prípadným zväčšením mapy) a jednak v natívnych súradniciach mapy Entropia Universe. Tento zdvojený kurzor je užitočný hlavne pri kalibrácii súradnicového systému. Tomu však bude venovaná kapitola ďalej v texte. Po ľavej hrane okna sú rozmiestnené tlačidlá, plniace funkcie najčastejších príkazov: Zvoliť kontinent, Pridať nález, Filter zobrazení, Nastavenie, Generovanie marching squares a Vyhľadávanie. V menu hlavného okna sú potom menej používané príkazy k importu a exportu dát.

Popis ďalších okien je uvedený v prílohe 1(Manuál). Popis hlavného okna tu je pre účel lepšieho pochopenia nasledujúceho textu, ktorá sa vzťahuje k programovaniu jeho funkcionality.

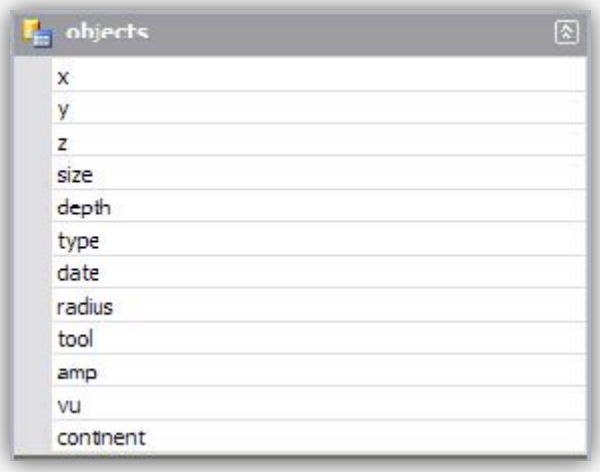
4.3 Hlavné okno pomocou Visual Studio 2008

Návrh okna začína príkazom pre vloženie nového objektu do projektu. Objektom bude okno systému WPF. Tím sa vytvorí prázdne okno, odpovedajúce xaml kódu a k nemu pridružený C# kód. Ovládacie prvky je možné do okna kresliť myšou, pričom sa generuje odpovedajúci xaml kód. Premiestňovanie elementov je veľmi podobné box modelu, ktorý môžeme vidieť na webe. Generátor kódu sa správa relatívne slušne, a však má sklon prichytávať pozíciu okna k nevhodnému okraju, poprípade urobiť veľkosť tlačidla závislú na veľkosti okna, čo väčšinou nie je to, čo by užívateľ chcel. Nie je problém si ručne opraviť xaml a potom sa vrátiť späť k designu, avšak Visual Studio si kód zasa prepíše podľa seba, hneď ako začneme s ovládacím prvkom znova pracovať.

Keďže celý proces návrhu okna je nápadne podobný návrhu okna v prostrediu NetBeans v jazyku Java, vo Visual Studio chýba výber udalostí pre jednotlivé ovládacie prvky. Dvojklikom na ovládací prvok sa väčšinou dostane do kódu nejaká jeho základná udalosť (napríklad pre tlačidlo je to udalosť onClick, pre okno je to onLoad), a však k ďalším udalostiam sa takto nedostaneme. Je potreba začať písať kód pre udalosť priamo v xaml, *intellisense* pritom navrhuje všetky možné vlastnosti elementov a medzi nimi i *handlery* udalostí. Medzi nimi si je potreba vybrať a nechať sprievodcu napísať hlavičku príslušnej funkcie. Je to vlastne krok späť. V celku podobným spôsobom bolo riešené pridávanie udalostí vo Visual Studiu verzia 6.0 v projektoch, používajúce systém MFC, ktorý je jedným z predchodcov WPF.

4.4 Uloženie dát

Dáta, ktoré sa v aplikácii ukladajú sú nálezy. Nálezy sú uložené v nešifrovanej databáze .NET SQL Server Compact 3.5. Tá je za účelom jednoduchšej manipulácie uložená ako súbor, nachádzajúci sa v adresári databáze. Dátové položky tabuľky sú zobrazené obrázku 4.1.



Obrázok 4.1

Prvé tri položky sú pritom súradnice nálezu v súradnicovom systému Entropia Universe, *size* je veľkosť nálezu, *depth* je hĺbka v akom sa nerasty nachádzajú, *type* je reťazec s menom suroviny, *date* je dátum ku ktorému bol nález zaznamenaný, *radius* súvisí s spôsobom zobrazovania nálezu pomocou Marching Sqares. *Tool* je meno nástroja, ktorým bol nález zaznamenaný a *amp* a *vu* sú parametry nálezu. *Continent* je meno kontinentu, kde bol nález zaznamenaný. Nálezy zo všetkých kontinentov sú v jednej tabuľke, oddeliť sa ich dá jednoduchým SQL príkazom *select.*

Nálezy sa musia zobrazovať na mapách. Mapy sú uložené v adresári *continents*. Každá mapa je uložená ako niekoľko súborov. Jednak je tu *názov-mapy.jpg*, obsahujúci vlastnú bitmapu, potom je tu *názov-mapy.csv*, obsahujúci informácie, nutné k prepočítaniu súradníc rastru na súradnice Entropia Universe a späť. Tretí nepovinný súbor *názov-mapy_calibrate_tp.csv*, obsahuje súradnice špeciálneho typu, *Teleport*. Pozície teleportov sú dobre známe aj zakreslené na mape a je preto vhodné ich využiť ku kalibrácii transformácie súradníc.

4.4.1 Transformácia súradníc

Jedná sa teda o transformáciu medzi súradnicami rastru a nativními súradnicami hry. Transformácia je daná štyrmi hodnotami, uloženými v .csv súboru, ktorá sprevádza mapu: *x*, *y*, *xoff* a *yoff*. Prvé dve sú šírka a výška mapy v natívnych súradniciach a ďalšie dve sú súradnice ľavého spodného rohu mapy. Súradnicový systém Entropia Universe má body (0,0) v ľavom spodnom rohu obrazovky,

pričom kladná polo os *x* smeruje vpravo a kladná polo os *y* smeruje hore. Transformácia súradníc v pixloch sa dá potom zapísať ako:

$$u = x\frac{s}{width} + xoff$$
$$v = y\frac{height - 1 - t}{height} + yoff$$

Kde *u, v* sú výsledné súradnice v priestore EU, *s, t* sú súradnice v pixloch a *width, height* je rozmer rastru v pixloch. Význam ostatných symbolov zostává rovnaký. Y-ová os je pritom prevrátená, keďže súradnicový systém Windows / DirectX má body (0,0) hore a kladná os *y* smeruje dolu. Táto transformácia sa dá zapísať pomocou matice:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = (s \;\; t \;\; 1) \cdot \begin{pmatrix} \frac{x}{width} & 0 & xoff \\ 0 & \frac{-y}{height} & y \cdot \frac{height - 1}{height} + yoff \\ 0 & 0 & 1 \end{pmatrix}$$

Spätná transformácia je potom daná jednoducho – inverznou maticou.

4.4.2 Postup kalibrácie mapy

Kalibrácia mapy prebieha v dvoch jednoduchých krokoch, ktoré sa prevedú postupne. Najprv pre horizontálne x-súradnice a potom analogicky aj pre y-súradnice. Keď nakopírujeme do adresára *continents* nový obrázok s mapou (napr. **Eudoria.jpg**), je potreba k nemu dodať súbor s odhadom kalibrácie (**Eudoria.csv**). Ten vyzerá takto:

```
x;y;xoff;yoff
30000;30000;0;0
```

Potom dodáme súbor so súradnicami teleportov (**Eudoria_calibrate_tp.csv**) a môžeme spustiť program. Po načítaní mapy (je nutné zmeniť kontinent, keď ho program nenájde sám) sa pozrieme na teleporty, ktoré sú zobrazené ako farebné body, rozmiestnené po mape. Pomocou kurzoru myši a súradníc pod ním zmeriame vzdialenosť v pixloch medzi najľavejším a najpravejším teleportom. Vzdialenosť bodov je 2120px - 270px = 1850px, pričom vzdialenosť odpovedajúcich značiek na mape je 2269px – 288px = 1981px. To znamená že teleporty sa oproti značkám rozbiehajú, čo môžeme pozorovať aj okom. Je preto potreba upraviť šírku mapy, a to prostým pomerom:

$$x' = x \cdot \frac{\Delta_{body}}{\Delta_{mapa}} = 30000 \cdot \frac{1850px}{1981px} = 28016$$

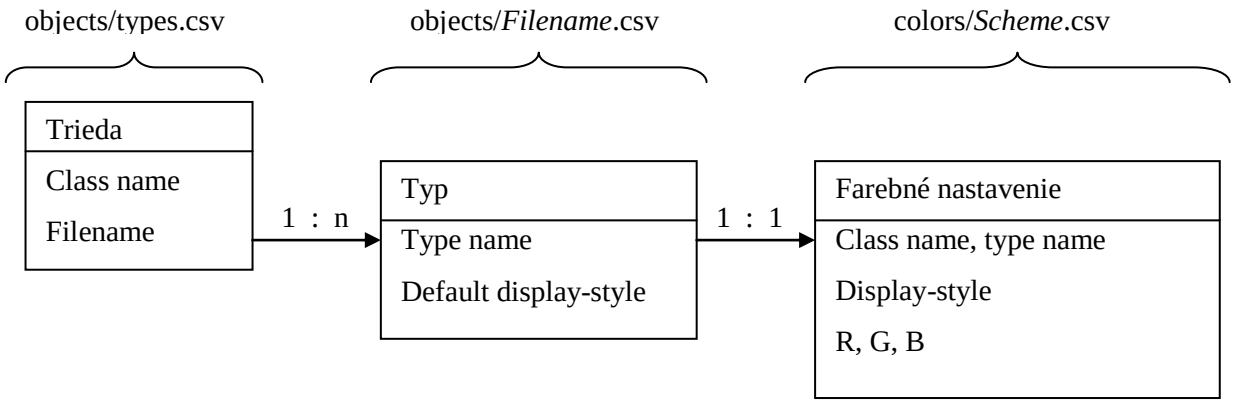
Keď pôvodnú hodnotou x(30000) nahradíme touto hodnotu x(28016) a znova spustíme program, teleporty sa oproti značkám na mape rozbiehať nebudú. Je však pravdepodobné, že budú trocha posunuté. Teraz môžeme znova myšou zmerať vzdialenosť medzi mapou a bodmi (teraz už len vodorovnú vzdialenosť ktoréhokoľvek teleportu od odpovedajúcej značky), napríklad 6856 a 6984

(teraz je v súradniciach mapy, nie v pixloch). Je teda zrejmé, že je potreba hodnoty posunúť o 128 jednotiek tak, že nastavíme xoff na 128 a tým je mapa nakalibrovaná horizontálne. Keď postup opakujeme a prevedieme analogickú vertikálnu kalibráciu, získame hodnoty, ktoré zaručia správne umiestnene teleportov a tým pádom i ostatných značiek na mape.

4.4.3 Uloženie typov objektov

Nastavenie programu je uložené v niekoľkých ďalších zložkách v súboroch s formátom csv. Farebné schémy sú v zložke *colors*, pritom *colors/default.csv* je prednastavená farebná schéma.

Typy objektov ako také sú členené do jednoduchej hierarchie: trieda objektu a typ objektu. Triedy sú uložené v zložke *objects* v súboru *types.csv*. Ten obsahuje zoznam názvov tried objektov a odpovedajúcich súborov kde sú definované typy súboru a prednastavený spôsob ich zobrazení. Nasledujúci diagram zhrnuje dátové schéma:



Display-style pritom nadobúda hodnotu *point* (bod), takto sú zobrazované napríklad teleporty alebo hangáre alebo *blob* (iso-povrchy), zobrazené pomocou marching squares.

4.5 Implementácia Marching Squares

Matematická báza pre algoritmus Marching Squares (MS) už bola čiastočne popísaná v kapitole 3 a preto ju nebudeme znova opisovať. Zamerajme sa hlavne na optimalizáciu algoritmu. Čo však v kapitole 3 nebolo, je generovanie vstupu pre MS a generovanie skalárneho pola.

V našej aplikácii nám ide o zobrazenie obláčikov okolo množiny bodov a ich automatické zlúčenie. Tomu sa slangovo hovorí „*metaballs*“. Každý bod našej množiny sa dá predstaviť ako nabitú časticu, ktorá emituje iso-hodnoty. Skalárne pole je potom sumou energií týchto žiaričov. V anglickom jazyku sa tomu hovorí „*attractors*“. Z fyziky vieme, že energia, s ktorou častica pôsobí je priamo úmerná štvorici ich vzdialeností:

$$F = \frac{Q_1 Q_2}{4\pi\varepsilon\varepsilon_0 r^2}$$

kde F je sila, s ktorou častice na seba pôsobia, Q_1 a Q_2 sú ich náboje a ε je permitivita prostredia v ktorom sú častice. Nepotrebujeme až tak veľmi úzkostlivo sledovať fyzikálne chovanie častíc, stačí nám základný princíp. Z rovnice sa dá ľahko odvodiť spôsob ako počítať hodnoty skalárneho pola. Predstavme si žiarič (*attractor*) ako jednu časticu, a daný bod pola, ktorý chceme vyhodnotiť ako druhú časticu. Náboje častíc budú jednotkovo rovnaké, tak isto ako permeabilita. Rovnica, ktorá je naším riešením má takýto tvar:

$$E_p = \sum_i \frac{e}{|q-p|^2}$$

Energia bodu p v skalárnom poli je suma štvorcov prevrátených vzdialeností všetkých žiaričov q od daného bodu mriežky p. Konštanta *e* je energia žiariča a v našej aplikácii bude mať vždy hodnotu 1.

Ako základná verzia bola vytvorená naivná verzia algoritmu marching squares v jazyku C++, ktorá bola nastavená na generovanie marching squares nad mriežkou s rozlíšením 1024x1024 a 2000 attractory rozmiestnené náhodne po mriežke s viac-menej rovnomerným rozložením. Potom bola aplikácia spustená a profilovaná s nasledujúcim výsledkom :

CS:EIP	Symbol + Offset	64-bit	Timer samples
0x401420	CMarchingRectangles::Solve		31413
0x4014d0	CMarchingRectangles::GeneratePolys		297

Obrázok 4.2

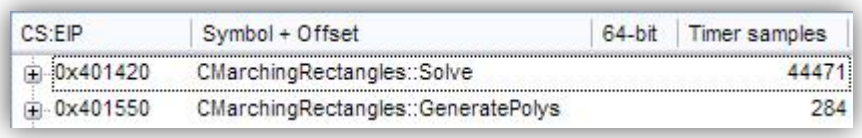
To znamená, že veľkú časť času spotrebuje funkcia Solve(), ktorá počíta iso-hodnoty na vrcholoch mriežky, čo sa dalo čakať. V podstate sa jedná o problém so zložitosťou: $O(m \cdot n^2)$, kde *m* je počet bobov, ktoré emitujú iso-hodnoty a *n* je veľkosť mriežky. Zanedbateľnú časť trávi v samotnom MS pri výpočte priesečníkov kontúr s mriežkou, zostavovaním geometrie a kreslením pomocou OpenGL.

Jednoduchou úpravou algoritmu môžeme znížiť efektívnu hodnotu *m*, a to zavedením maximálnej chyby. Maximálna chyba nám umožní spočítať vzdialenosť pri ktorom sa už nemusí počítať s daným žiaričom. Tým pádom sa už nemusia zúčastniť všetky body skalárneho pola, ale len tie, ktoré sú dosť blízko. Pre nejakú relatívnu chybu sa vzdialenosť spočíta takto:

$$err_{abs} = treshold \cdot err_{rel} \cdot m$$

$$d_{max} = \sqrt{\frac{1}{err_{abs}}}$$

Kde *m* je stále počet žiaričov, err_{rel} je maximálna prípustná relatívna chyba, *treshold* je prah na ktorom ležia iso-kontúry. Potom err_{abs} je maximálna prípustná absolútna chyba a d_{max} je maximálna vzdialenosť, do ktorej sa bude započítavať príspevok daného žiariča. Znova profilujeme ten istý test s prípustnou chybou 1%(keď sa povolí chyba 10%, tak je to na výsledku dobre vidieť a to je neprípustné) Výsledky sú zdrvujúce:



CS:EIP	Symbol + Offset	64-bit	Timer samples
0x401420	CMarchingRectangles::Solve		44471
0x401550	CMarchingRectangles::GeneratePolys		284

Obrázok 4.3

Test bol o 40% pomalší, ako pred tým. To sa dá vysvetliť veľkým množstvom *attraktorov*, ktoré sú dosť blízko. Keby bol charakter ich rozložení v ploche iný (viac samotných zhlukov), dalo by sa počítať s pozitívnym výsledkom. Ani zvýšenie maximálnej chyby na 10% neprinieslo pozitívne výsledky. Táto optimalizácia je totiž v našom prípade nepoužiteľná.

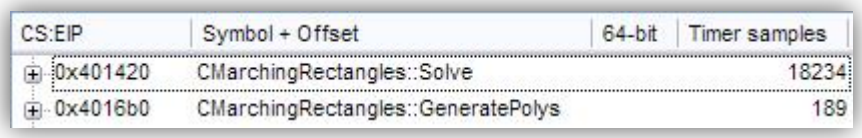
Ďalšou možnosťou ako optimalizovať v obmedzení počtu navštívených bodov v skalárnom poli, teda znížením n v $O(m \cdot n^2)$. Ideálne by bolo, keby sa nám podarilo navrhnúť algoritmus, ktorý by vyhodnocoval len tie body mriežky, kde hodnota skalárneho pola prekročí prah iso-kontúr a bezprostredne susediace body. Tieto body sú nutné pre určenie priesečníkov hodnoty pola s prahom. Keby nám stačilo generovať kontúry bez výplne, nebolo by nutné ani navštevovať ani body mriežky, kde všetky okolité body majú hodnotu pola väčší ako je prah (teda na nich nevznikajú žiadne kontúry).

Z vyplňujúcich algoritmov sa nám ponúkne semienkové vyplňovanie, ktoré má jednoduchú implementáciu a zároveň nemá zlé výsledky.

Postupujeme nasledujúcim spôsobom: každý vrchol mriežky má teraz svoj stav návštevnosti:

- nenavštívený
- naplánovaný k navštíveniu
- navštívený a vyhodnotený

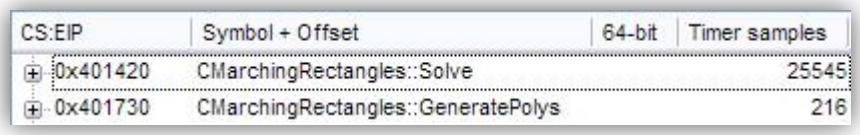
Na začiatku sa stavy všetkých bodov nastavia na nenavštívené. Potom sa pre každý *attraktor* nájde najbližší bod v mriežke a keď ešte nebol navštívený a ani naplánovaný k navštíveniu, tak sa vyhodnotí hodnota skalárneho pola na jeho pozícii (pričom sa započítavajú príspevky všetkých *attraktorov*, nielen toho aktuálneho). Keď je hodnota menší ako prah, už to ďalej nepokračuje. Keď je väčšia, všetky okolité body mriežky sa pridajú do fronty. Označí sa ako *naplánovaný k navštíveniu* (teda keď ešte nie je označený) a pokračuje tým istým spôsobom s vrcholným bodom fronty, až kým fronta nebude prázdna. Týmto sa zaručí, že všetky body mriežky pod ktorými by mohli byť iso-povrchy boli navštívené a algoritmus je teda kompletný. Zrýchlenie algoritmu silne závisí na pokrytej ploche mriežky. Ak je celá mriežka pokrytá iso-povrchmi, tak by bolo treba prejsť všetky body mriežky. Algoritmus tak prinesie len zvýšenú réžiu na výpočet. Profilovanie potvrdilo úspech optimalizácie:



CS:EIP	Symbol + Offset	64-bit	Timer samples
0x401420	CMarchingRectangles::Solve		18234
0x4016b0	CMarchingRectangles::GeneratePolys		189

Obrázok 4.4

Nárast rýchlosti je skoro 50%, čo nie je zlé. Viac optimalizačných metód už nemáme k dispozícii, takže sa budeme musieť zmieriť s výsledkom. Test so zavedením maximálnej chyby a zamietnutím *attraktorov*, ktoré sú za maximálnou vzdialenosťou aj v tomto prípade skončil neúspechom:



CS:EIP	Symbol + Offset	64-bit	Timer samples
0x401420	CMarchingRectangles::Solve		25545
0x401730	CMarchingRectangles::GeneratePolys		216

Obrázok 4.5

V tomto prípade je algoritmus pomalší zhruba o 38%, čo je zrovnateľný s predchádzajúcim štyridsať pri jednoduchej verzii bez semienkového vyplňovania.

V prílohe 3 (Benchmark) sa nachádza obrázok, ktorý zobrazuje na akých dátach boli algoritmy testované.

4.5.1 Zobrazenie marching squares vo WPF formulári

K zobrazeniu MS na formulári je využitá technika, ktorá je pre WPF pre kreslenie doporučená. Keď porovnávame s C++, nemáme tu žiadny ekvivalent OnPaint, ale polygóny sú prevedené na objekty.

Medzi výhody patrí tiež vlastnosť, že nie je potreba vytvárať dátové štruktúry, do ktorých by sa ukladali geometrie polygónov. Stačí umiestniť tvary na formulár tak, aby ich rodičom bol obrázok obsahujúci mapu. Ostatné veci za nás urobí WPF.

Napriek tomu je potrebné generovať MS v podstate dvakrát. Raz treba spočítať výplne (polygóny) a v druhom kroku treba spočítať kontúry. Sú teda dve tabuľky s iso-ohraničeniami.
WPF totiž automaticky vyhladzuje hrany všetkých svojich objektov.

Jedným z ďalších možných riešení problému bolo pospojovať polygóny tak, aby každá oblasť bola tvorená len z jedného polygónu. V tomto prípade by ale niektoré polygóny museli obsahovať diery a náročnosť výpočtu úlohy by týmto neúmerne vzrástla.

5 Záver

Úlohou mojej bakalárskej práce bolo vytvoriť software na zjednodušenie katalogizovania nálezov z hry Entropia Universe s následným grafickým zobrazením výsledku.

Pri testovaní na užívateľoch bol zistený pozitívny výsledok čo sa týka ovládania, prehľadnosti a manipulácie s bodmi. Rýchlosť chodu aplikácie je závislá na rýchlosti používaného počítača. Je potrebné mať príslušné knižnice (DLL súbory) prostredia WPF. Hlavným prínosom softvéru je väčší úspech hráčov pri hľadaní a nájdení surovín v hre s pomocou nanovo získaných výsledkov z programu. Užívatelia sa pomocou programu ľahšie zorientujú na obrovských mapách sveta.

Z ďalšieho pohľadu vývoja projektu sú tu dve cesty. V softvéry by bola vítaná funkcia OCR (Optical character recognition). S touto funkciou by sme si vedeli zjednodušiť zadávanie bodov do programu a to tak, že by sme za behu hry zo *screenshotu* získali všetky informácie potrebné na zadanie nového bodu do databázy. Ďalšou cestou vývoja je vytvorenie centrálnej databázy na internete, do ktorej by sa ukladali body. Užívatelia by nahrávali vlastné nálezy na centrálny server, kde by sa body zlúčili a dali by sa to opätovne stiahnuť. Získali by sme viac a presnejšie informácie o oblastiach.

Literatúra

[1] Dej a základné informácie o hre. Dokumenty dostupné na URL www.entropiauniverse.com (február 2008)

[2] Recenze hry Entropia Universe, MMORP info z online her. Dokument dostupný na URL http://mmorpg.onlajnovky.cz/entropia-universe/2008/01/30/ (február 2008)

[3] Windows Presentation Foundation. Dokument dostupný na URL http://msdn.microsoft.com/en-us/library/ms754130.aspx (február 2008)

[4] Algorithmus Marching Squares. Dokument dostupný na URL http://en.wikipedia.org/wiki/Marching_squares (február 2008)

[5] Visual Studio 2008: Prehľad. Dokument dostupný na URL http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx (február 2008)

[6] Rephael Wenger: Isosurfaces: Geometry, Topology and Algorithms, unpublished manuscript;

[7] Bhaniramka, Wenger, Crawfis: Isosurfacing in Higher Dimensions, Vis ’00, 2000;

[8] Hans-Jochen Bartsch, Matematické vzorce, štvrté vydanie, 2006;

Zoznam príloh

Príloha 1. Manuál

Príloha 2. Zdrojové texty

Príloha 3. Benchmark

Príloha 4. Obsah CD

Príloha 5. CD

Príloha 1: Manuál

1 Inštalácia

1.1 Systémové požiadavky

- **Podporované operačné systémy (OS):** Windows Server 2003; Windows Server 2008; Windows Vista; Windows XP
- **Procesor (CPU):** 400MHz Pentium procesor alebo ekvivalentný (minimálne); 1GHz Pentium procesor alebo ekvivalentný (doporučené)
- **Pamäť (RAM):** 128MB (minimálne); 256 MB (doporučené)
- **Pevný Disk (HDD):** Viac ako 20 MB voľného miesta
- **Obrazovka (Display):** 800x600, 256 farieb (minimálne); 1024x768 high color, 32-bit (doporučené)

1.2 Požadované softwarové vybavenie

- Microsoft .NET framework 3.5
- Microsoft Windows Installer 3.1
- Microsoft SQL Server compact 3.5

Software potrebný pre spustenie programu môžete nainštalovať predne zo zdrojového CD, alebo cez sprievodcu inštalácie prostredníctvom Internetu.

1.3 Inštalácia

Pre nainštalovanie programu treba spustiť ***setup.exe*** a ďalej sa riadiť pokynmi inštalátoru.

1.4 Odinštalácia

V prípade odinštalácie programu treba spustiť ***Control Panel*** systému Windows, nasledovne vybrať položku ***Pridať či odobrať programy,*** vybrať program, ktorý chceme odinštalovať a kliknúť na tlačítko ***Odobrať.***

2 Základný popis

2.1 Nález

Pred použitím programu musíme poznať pár pojmov, aby sme vedeli čo od nás program na niektorých miestach vyžaduje. Medzi tieto pojmy patrí aj nález. Pod nálezom po anglicky: „find“ rozumieme informácie, ktoré sú spoločné s nájdenou surovinou v hre na jednom mieste. K nálezu patriace informácie nájdeme v nasledujúcej tabuľke s popisom:

Continent*	Kontinent v hre na ktorom sme surovinu našli
X*	Súradnica X suroviny
Y*	Súradnica Y suroviny
Z	Výska v ktorom sme surovinu našli, doplňujúca informácia
Depth*	Hĺbka v ktorom sme surovinu našli
Type*	Typ suroviny
Size*	Veľkosť nálezu
Date / Time	Dátum a Čas nálezu
Tool	Názov prístroja s ktorou sme surovinu našli. Doplňujúca informácia, nastavuje sa predne v nastaveniach
Radius	Vzdialenosť hľadania prístroja TOOL. Doplňujúca informácia
Amp	Použitý zosilňovač pri hľadaní. Pripína sa k prístroji TOOL, zvyšuje efektívnosť. Doplňujúca informácia, nastavuje sa predne v nastaveniach
V.U.	Verzia hry pri nálezu. Doplňujúca informácia, nastavuje sa predne v nastaveniach

*Hlavné informácie získané z hry pri nálezu.

2.2 Spustenie programu

Program sa spúšťa pomocou ikony na pracovnej ploche, ktorý bol pridaný prostredníctvom inštalácie. Ďalšia možnosť spustenia programu je cez programový adresár prostredníctvom spúšťacieho súboru Euram.exe.

3 Popis software

3.1 Okná programu

1) Nástrojový panel, ktorý obsahuje položky: **Clear All** – vymazanie databázy s všetkými nálezmi na všetkých kontinentoch. **Import** - pridanie nálezov zo súboru typu .XML alebo .CSV. **Export** – uloženie databáze do súboru .XML alebo .CSV.
2) **Change Continent** – zmena kontinentu.
3) **Add Find** – pridanie nálezu do databáze (na mapu).
4) **Filter** – Filtrovanie bodov a oblastí na mape. Cez túto položku sa dá nastaviť ktoré typy surovín sa majú zobraziť na mape.
5) **Properties** – Nastavenia programu.

6) **Generate** – Spustiť výpočet pre vytváranie oblastí.
7) **Search** – Táto funkcia slúži na vymazanie bodov. V prípade, že sme pri zadávaní urobili chybu alebo je bod už neplatný.
8) Informácia o aktuálnej pozície kurzoru mysy na mape. Prvá súradnica je aktuálny pixel obrázku a druhá súradnica je aktuálna pozícia na mape v hernom svete.
9) **ZOOM** – je to slider na približovanie bodu na mape.
10) Hlavná mapa, sem sa vykresľuje výsledok.

3.1.1 Myš

1) Po stlačení a držaní ľavého tlačidla na myši na mape sa vieme pohybovať po mape
2) Po stlačení pravého tlačidla pridáme bod na aktuálnu pozíciu kurzora.
3) ZOOM – Približovanie a vzdialenie mapy

3.1.2 Change Continent

Okno – Change Continent

Pomocou tohto okna vieme prepínať medzi kontinentmi. Kliknutím na tlačidlo **Load** načítame požadovaný kontinent.

3.1.3 Add Find

Okno slúžiace na zadanie konkrétneho nálezu. Toto okno sa nám objaví aj prípade stlačenia pravého tlačidla na myšy, pri ktorom sa automaticky vyplnia dve položky a to **X** a **Y** súradnice. V opačnom prípade si ich musíme zadať sami. Súradnice sa získajú z nálezového papiera, ktorý hráč automaticky

dostane v hre prípade, že nájde surovinu. Na nálezovom papieri sa nachádzajú aj informácie o hĺbke, veľkosti a typu. Kliknutím na tlačidlo **ADD** pridáme nález.

Okno - Add Find

3.1.4 Filter

Okno – Filter

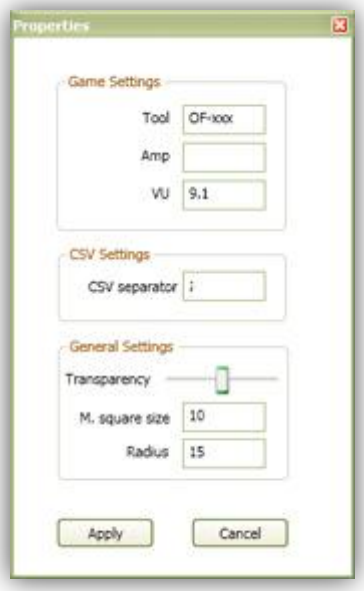
Okno slúžiace na nastavenie farebných schém vykresľovania oblastí a bodov na mape. Položky sú rozdelené do troch kategórií. Prvé dve (Energy material a Ore) sú oblasti. K tretej položke (Other) patria statické body, ktoré sú vynechané z výpočtu oblastí. Každá položka má svoju vlastnú farbu, ktorú si môžeme podľa seba zmeniť. Zaškrtávací štvorec slúži pre pridanie a odobranie vybraného typu do výpočtu. Po nastavení vlastných farieb je možné uložiť schémy do súboru a opätovne načítať.

3.1.4.1 Color Scheme

Okno – ColorScheme

- Load – Načítanie
- Save – Uloženie
- Random – Náhodné farby
- Cancel – Zrušiť

3.1.5 Properties



Okno – Properties

Tool – Názov použitého prístroja pri nálezu
Amp – Názov použitého zosilňovača pri nálezu
VU – Verzia hry pri nálezu

CSV separátor – Oddeľovač v súboroch .csv pri načítaní a ukladaní

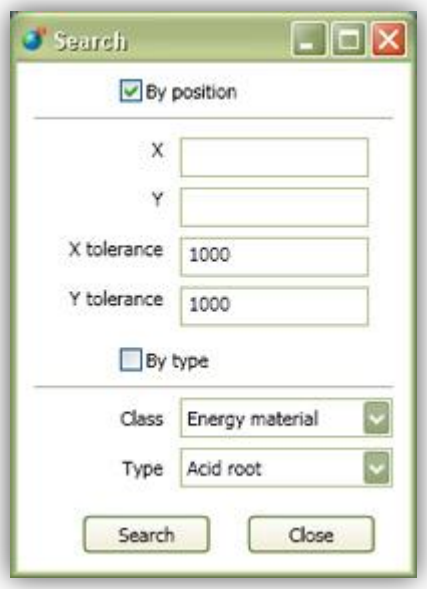
Transparency – Priehľadnosť vykreslených oblastí
Min square size – Presnosť vytváraní polygónov pri výpočte
Radius – Určuje hranicu od ktorej má oblasť rozdeliť na dve časti podľa vzdialenosti dvoch bodov.

Apply – tlačidlo pre uloženie zmien.

3.1.6 Search

Doplňujúca funkcia programu, ktorá je určená pre vymazanie nadbytočných alebo neplatných bodov. Cez túto funkciu sa dá nájsť bod v databáze a nasledovne vymazať. Možnosti vyhľadávania sú:

- Podľa súradnice – Zadáme približnú súradnicu a rozmedzie v rámci ktorých sa má bod hľadať.
- Podľa typu – Zadáme kategóriu a typ suroviny
- Súčasne podľa oboch kritérií

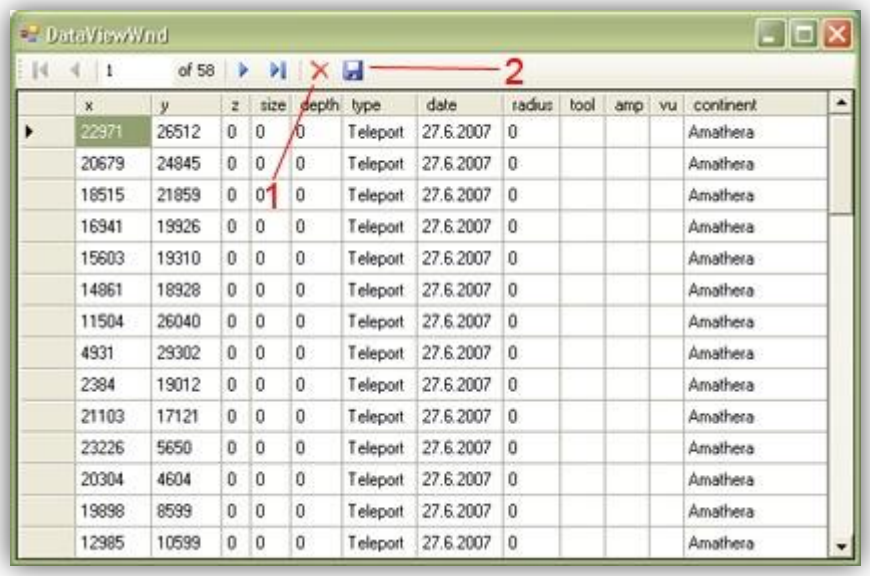


Okno – Search

3.1.6.1 Search result

Okno slúžiace na zobrazenie výsledkov vyhľadávania.

1) Vymazanie označenej položky
2) Uloženie zmien do databáze



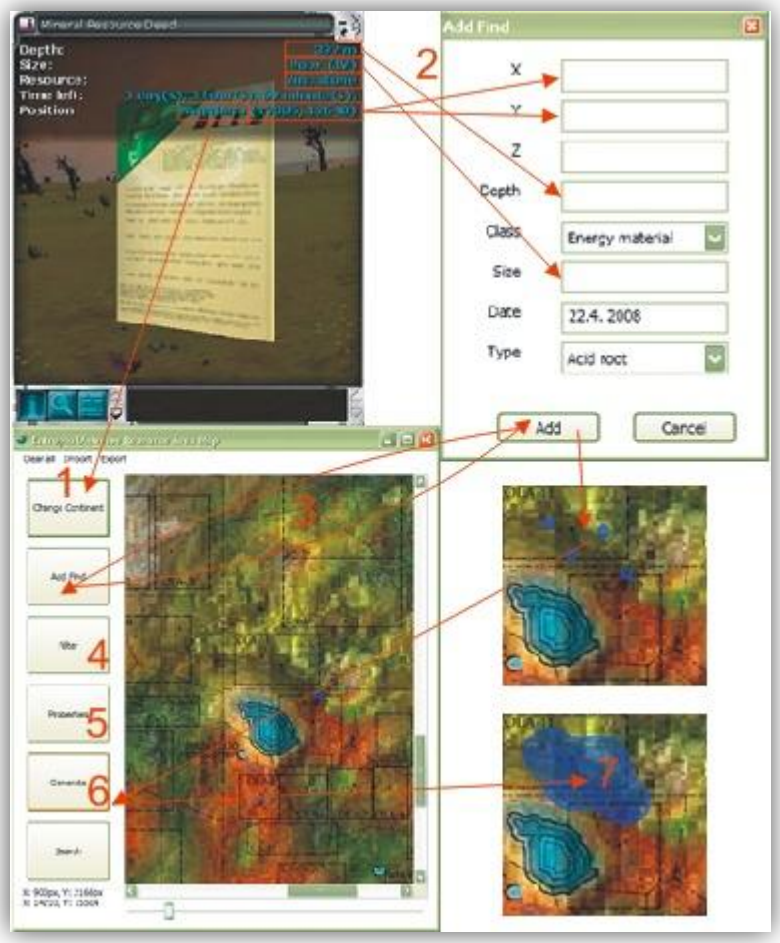
x	y	z	size	depth	type	date	radius	tool	amp	vu	continent
22971	26512	0	0	0	Teleport	27.6.2007	0				Amathera
20679	24845	0	0	0	Teleport	27.6.2007	0				Amathera
18515	21859	0	0	0	Teleport	27.6.2007	0				Amathera
16941	19926	0	0	0	Teleport	27.6.2007	0				Amathera
15603	19310	0	0	0	Teleport	27.6.2007	0				Amathera
14861	18928	0	0	0	Teleport	27.6.2007	0				Amathera
11504	26040	0	0	0	Teleport	27.6.2007	0				Amathera
4931	29302	0	0	0	Teleport	27.6.2007	0				Amathera
2384	19012	0	0	0	Teleport	27.6.2007	0				Amathera
21103	17121	0	0	0	Teleport	27.6.2007	0				Amathera
23226	5650	0	0	0	Teleport	27.6.2007	0				Amathera
20304	4604	0	0	0	Teleport	27.6.2007	0				Amathera
19898	8599	0	0	0	Teleport	27.6.2007	0				Amathera
12985	10599	0	0	0	Teleport	27.6.2007	0				Amathera

Okno – Search Result

3.2 Použitie

Popis používania programu:

1. Získame z hry potrebné informácie pre pridanie bodu do databázy.
2. Zadáme získané informácie do programu
3. Body 1 a 2 opakujeme kým nemáme dostatočný počet nahodených bodov pre vykreslenie oblastí
4. V **Properties** nastavíme prístroj, zosilňovač a verziu hry
5. Vo **Filter** nastavíme ktoré typy chceme zobraziť na mape, prípadne aj farby.
6. Spustíme výpočet a vykresľovanie.
7. Z mapy si prečítame získané informácie.



Príloha 2: Zdrojové texty

3.3 Trieda nálezu

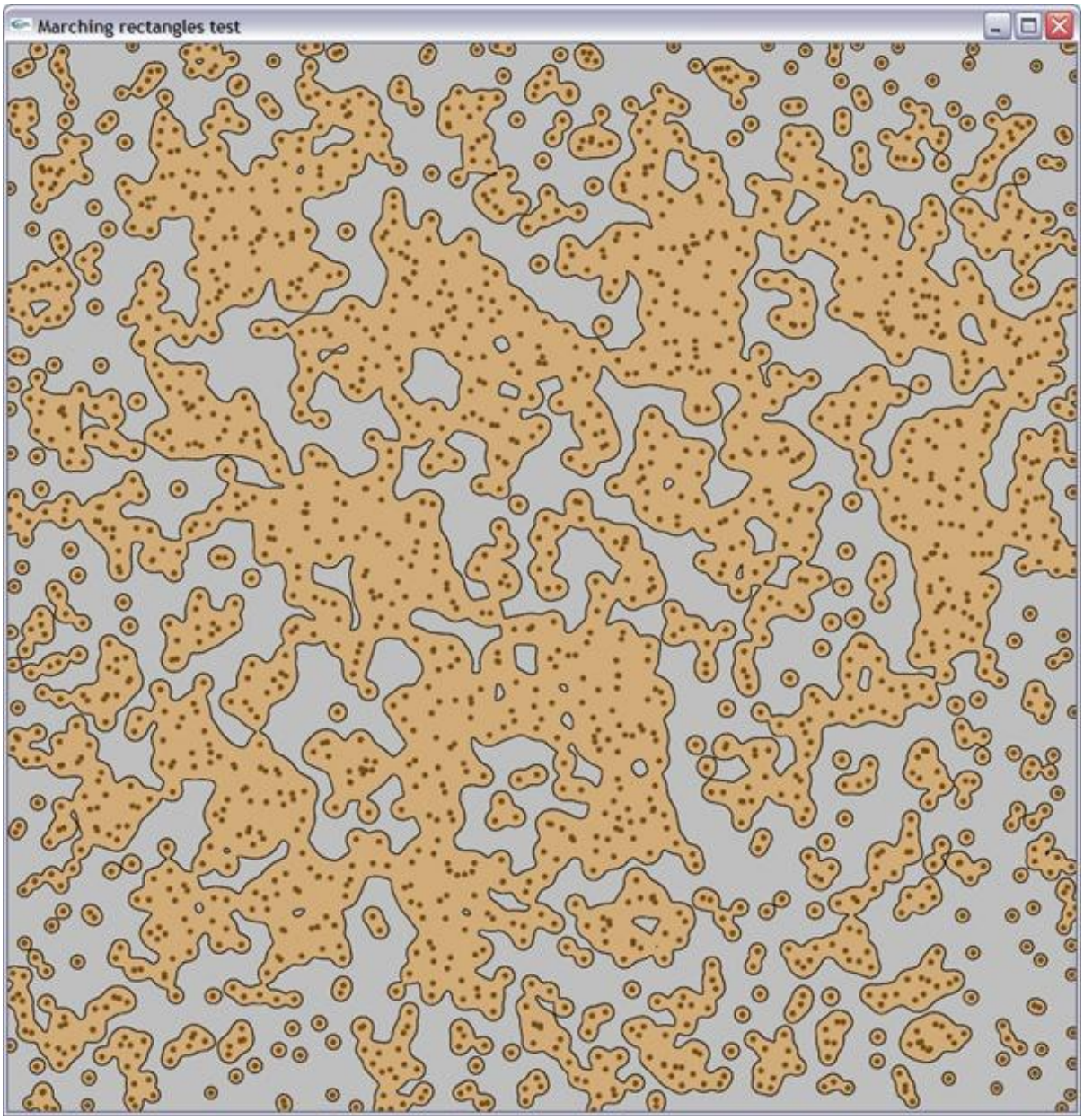
```
public class TFind
    {
        public int x, y, z;
        public int size;
        public int depth;

        public String type; // ore type / energy matter type / outpost /
teleport

        public int mm, dd, yyyy;

        public int radius;
        public String tool, amp, vu;

        public TFind(int _x, int _y, int _z, int _size, int _depth,
            String _type, long datetime)
            : this(_x, _y, _z, _size, _depth,
            _type, 0, 0, 0, 0, "", "", "")
        {
            DateTime dt = new DateTime(1970, 1, 1, 0, 0, 0);
            dt = dt.AddSeconds(datetime);
            dd = dt.Day;
            mm = dt.Month;
            yyyy = dt.Year;
        }

        public TFind(int _x, int _y, int _z, int _size, int _depth,
            String _type, int _mm, int _dd, int _yyyy)
            : this(_x, _y, _z, _size, _depth,
            _type, _mm, _dd, _yyyy, 0, "", "", "")
        { }

        public TFind(int _x, int _y, int _z, int _size, int _depth,
            String _type, int _mm, int _dd, int _yyyy, int _radius, String
_tool,
            String _amp, String _vu)
        {
            x = _x;
            y = _y;
            z = _z;
            size = _size;
            depth = _depth;
            type = _type;
            mm = _mm;
            dd = _dd;
            yyyy = _yyyy;
            radius = _radius;
            tool = _tool;
            amp = _amp;
            vu = _vu;
        }

        public long n_Date()
```

```
        {
            DateTime dt = new DateTime(1970, 1, 1, 0, 0, 0);
            DateTime date = new DateTime(yyyy, mm, dd, 0, 0, 0);
            TimeSpan span = date.Subtract(dt);
            return (long)span.TotalSeconds;
        }
    }
```

3.4 Triedy bodov pre marching squares

```
class TVertex2
    {
        public enum Status {
                Unvisited,
                Scheduled,
                Visited
          };

        public Vector2f v_pos;
        public float f_energy;
        public Status n_visited;

        public TVertex2()
          {}

        public TVertex2(Vector2f _v_pos)
          {
            v_pos = _v_pos;
            f_energy = 1;
            n_visited = Status.Unvisited;
          }

        public TVertex2(Vector2f _v_pos, float _f_energy)
          {
            v_pos = _v_pos;
            f_energy = _f_energy;
            n_visited = Status.Unvisited;
         }


class TVertex
    {
        public Vector2f v_pos;
          public float f_energy;

        public TVertex()
          {}

        public TVertex(Vector2f _v_pos)
          {
            v_pos = _v_pos;
            f_energy = 1;
          }

        public TVertex(Vector2f _v_pos, float _f_energy)
          {
            v_pos = _v_pos;
            f_energy = _f_energy;
          }
```

Príloha 3: Benchmark



Benchmark: grid 1024x1024, 2000 atraktorov

Polygóny sú kreslené priehľadne. Body značiace *atraktory* sú v skutočnosti čierne, ale sú prekryté práve béžovými priehľadnými polygónmi. Žiadnemu *aliasingu* nedochádza.

Príloha 4: Obsah CD

Src – Zdrojové súbory projektu v jazyku C# pre aplikáciu Visual Studio 2008

Bin – Binárne súbory projektu, spustiteľná inštalácia aplikácie pod Windows XP

Prereq – Pre rekvizity. Binárky potrebných knižníc pre spustenie programu pod Windows XP

Technická sprava – Elektronická verzia technickej správy vo formáte .docx (MS WORD 2007) a .pdf (Adobe portable document format).